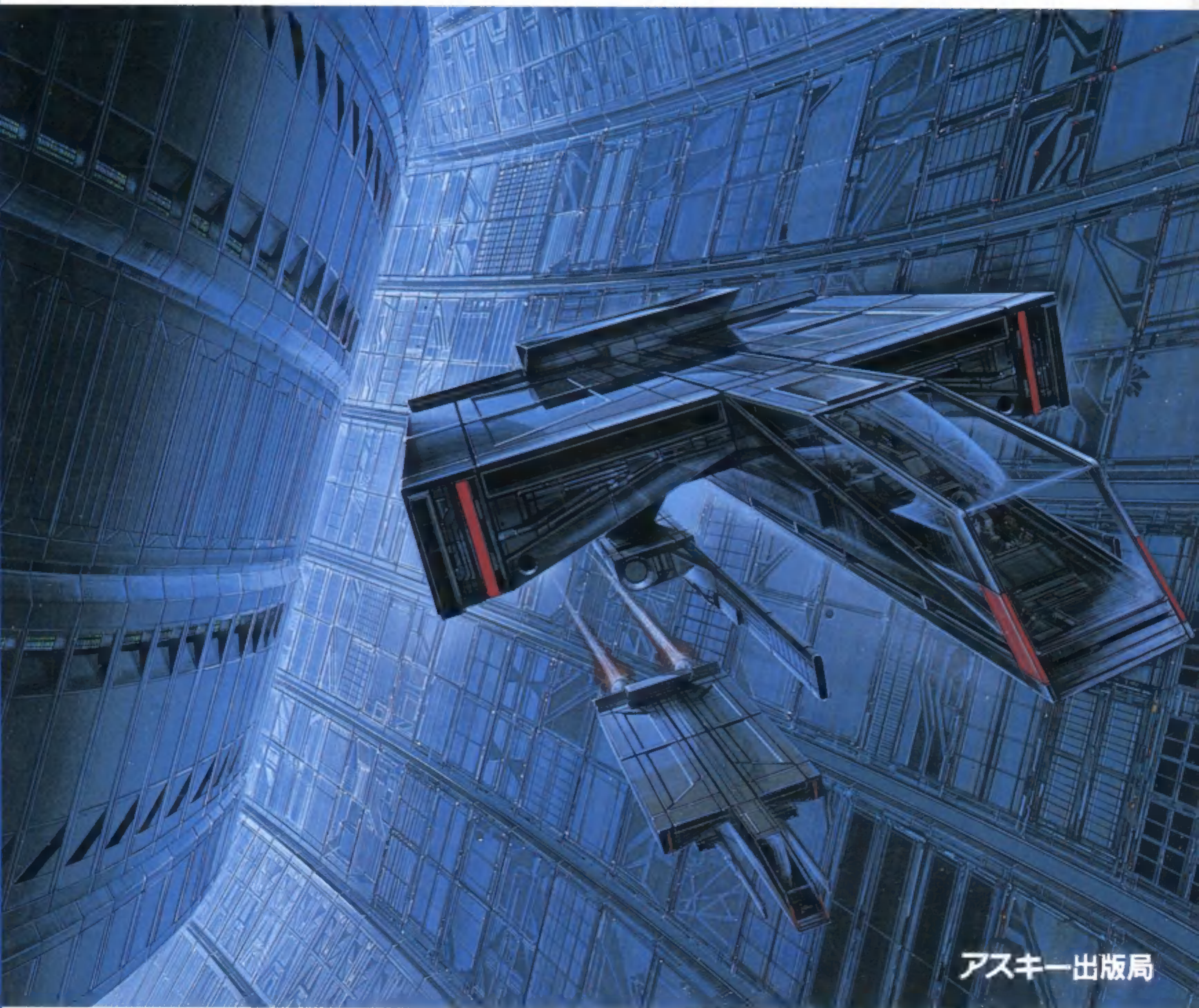


PC-8801mkⅡSR

マシン○ゲーム プログラミング

日高 徹 著



アスキー出版局

PC-8801mkⅡSR

マシン語ゲーム プログラミング

日高 徹 著

アスキー出版局

PC-8801mkⅡSR

マシン語ゲーム プログラミング

日高 徹 著

アスキー出版局

本文デザイン・守屋一於
本文イラスト・児玉敏彦

はじめに

本書は、ずばりマシン語でリアルタイムゲームを作りたいあなたに贈る、マシン語ゲーム製作『種あかしの秘本』です。この1冊の本の中に、マシン語ゲーム作りのノウハウがぎっしりと詰まっています……。

「究極の8ビットマシン」といわれてきたNECのPC-8801シリーズですが、SRの登場でその地位はますます揺るぎないものになったといえそうです。そして、この人気の一翼を担っているのは、紛れもなくマシン語ゲーム・ソフトなのです。しかし、本体付属のマニュアルは、BASICに関しては大変わかりやすく、またていねいに書かれているのですが、マシン語についてはどういうわけか、できることならやらないで欲しい、といわんばかりの内容でしか書かれていません。そのため、マシン語をマスターしたい人は、どうしても市販の書籍に頼らざるを得ないことになります。

マシン語に関する書籍はかなり多くあり、また内容的にも立派なものばかりなので、ケチなどつけようがありませんが、読んでみると読者の要求とはかなり違っているような気がするのです。つまり、マシン語は教えてくれても、ゲームのためのマシン語は教えてくれないのです。禅問答のような気がするかもしれませんが、入門者にとってこのギャップは実に大きな障害であり、ここで挫折していった人のことを思うと残念でなりません。

本書は、ゲームのためにマシン語をマスターしたいあなたが、回り道をすることなく目的を達成できるよう、マシン語の基礎からソフトハウス向けの超高度なテクニックまで、そのすべてをわかりやすく解説したものです。その上、ゲームが目的ですが、結果を画面で確かめながらマシン語をマスターできるため、マシン語の実践的な入門書としても十分な内容となっております。また、マシン語プログラム製作の必需品であるアセンブラも、本書用にオリジナルのものが提供されています。これらは、これまで閉ざされてきた暗闇のゲームマシン語の世界に、最初から明かりをつけて、そのすべてを公開しようという、アスキーならではの大胆かつ雄大な企画なのです。

なお、本書執筆にあたり、心よくプログラム作成に協力をしてくださいました石塚圭樹氏、大熊英男氏、ならびにMF-ASMの掲載をこころよく了承してくださった藤井敬雄氏に対し、この誌面を借りまして厚く御礼申し上げる次第です。

Contents 1

CHAPTER 1 ●ウォーミング・アップ

1. 小道具…これだけはそろえておこう! —— 20
2. 数…二進数と十六進数について —— 21
3. アセンブラ…マシン語開発ツール —— 24
4. メモリーマップ…ハードウェアについて —— 26
5. 命令…ニーモニックとレジスタ —— 29
6. プログラム…その作成と実行 —— 31

CHAPTER 2 ●キャラクタ・パターンの表示と移動

1. 座標…ゲームのためのゲーム座標 —— 38
2. 豆腐…とりあえず白い四角形を表示 —— 40
3. パターン…キャラクタの作成 —— 46
4. パターン表示…キャラクタ登場 —— 49
5. パターン消去…キャラクタを動かす前に —— 57
6. パターン移動…データにそって移動 —— 64
7. 大量出現…1人じゃつまんない! —— 67
8. キー入力…コントロール&ショット —— 73

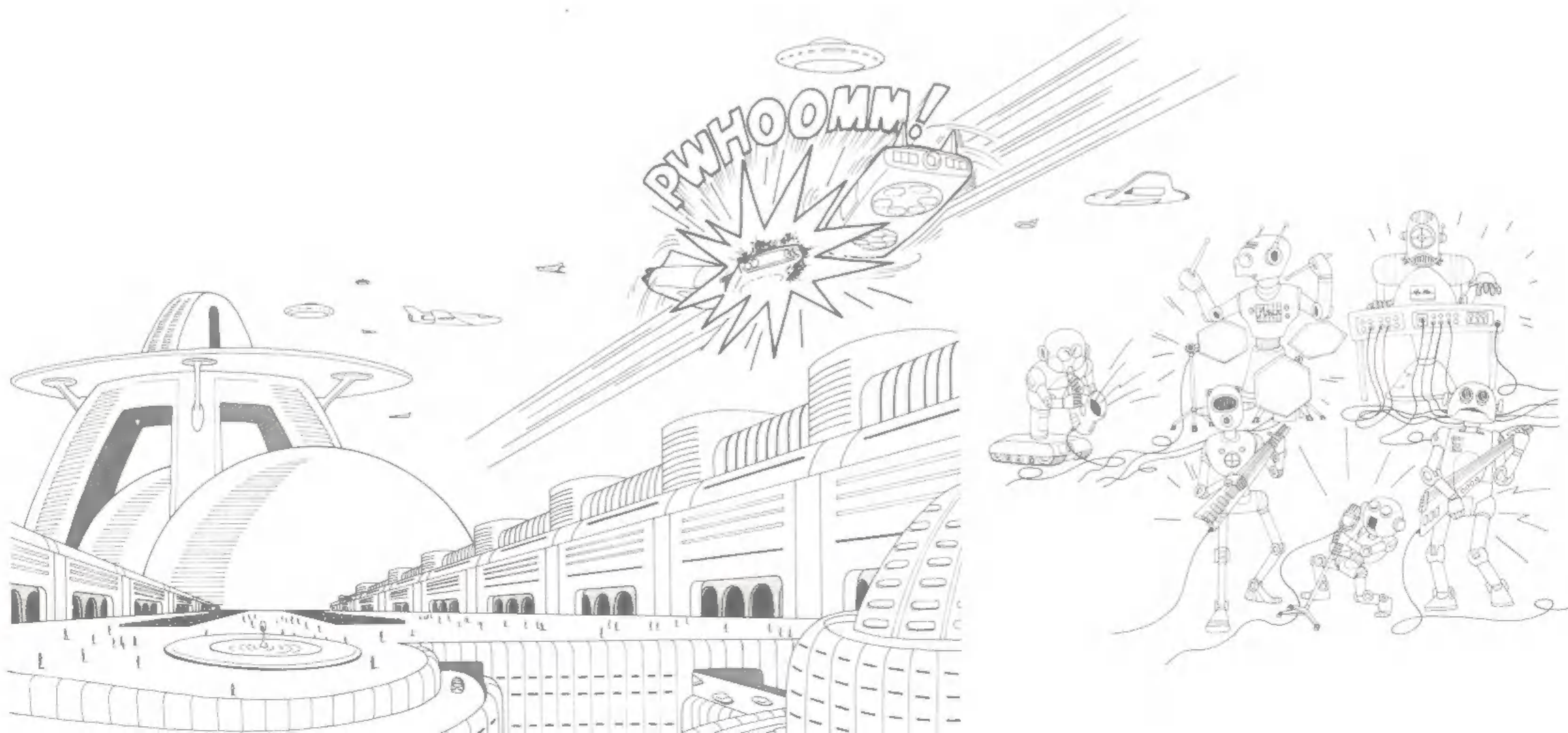


CHAPTER 3 ● 衝突と得点計算

1. 衝突の判定…ゲーム座標を用いる —— 84
2. 数字…文字と数字パターンの作製 —— 88
3. 計算…得点の計算と表示 その1 —— 93
4. BCD…得点の計算と表示 その2 —— 97
5. 衝突の処理…ゲームらしさの追求 —— 102

CHAPTER 4 ● 音楽演奏と効果音

1. BEEP音…音の仕組みとハードウェア —— 114
2. 音楽…BEEP 音楽用音程データ —— 118
3. 臨場感…BEEP による効果音 —— 122
4. FM音源とPSG…PC 8801 mk II SR 専用 —— 124
5. ミュージック…FM音源でハープシコード —— 130



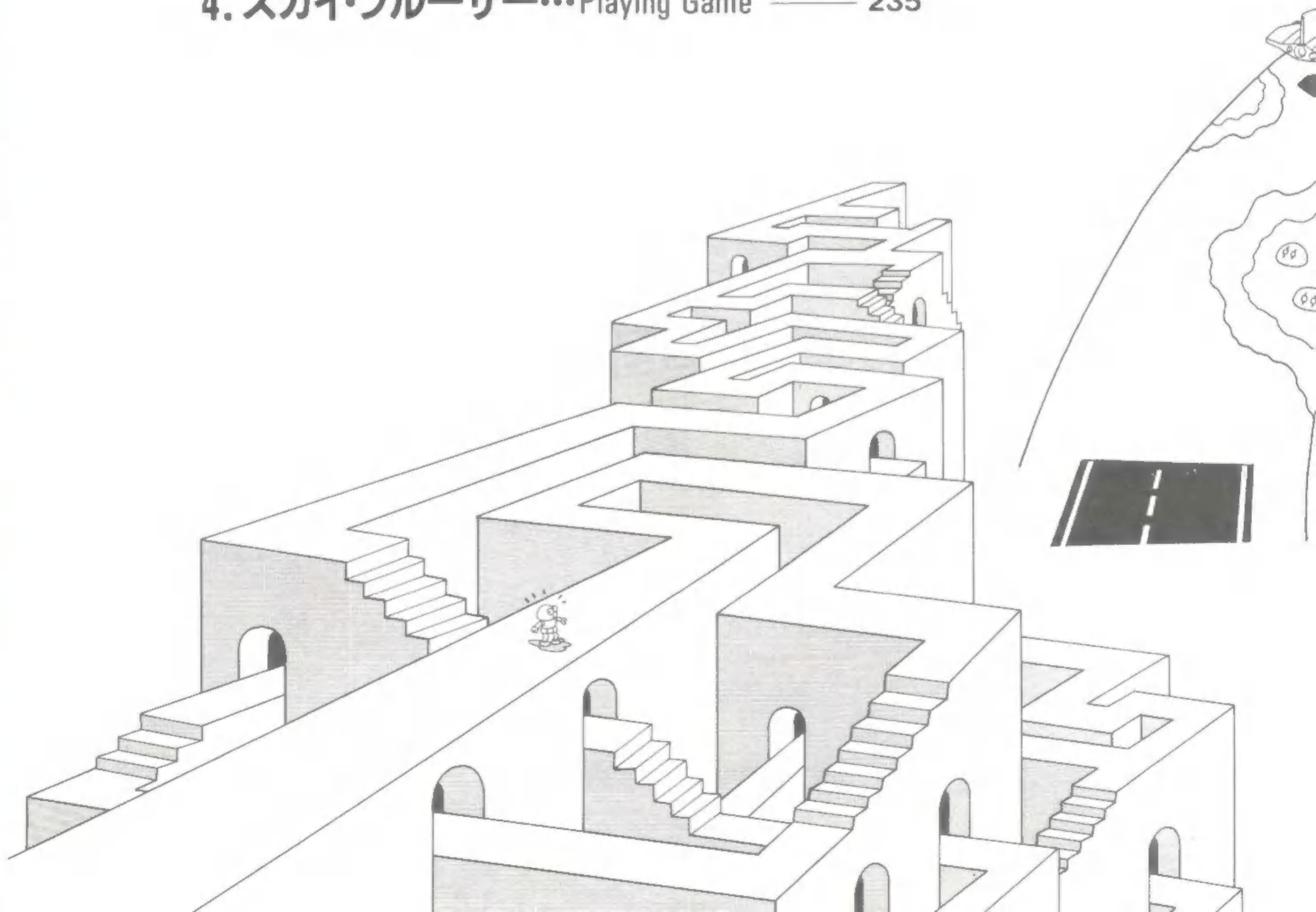
Contents ————— 2

CHAPTER 5 ● 迷路型ゲーム

- 1. 座標データ…行ける? 行けない? ——— 140
- 2. 圧縮…座標データのデータ量 ——— 142
- 3. キー入力…操作性の向上 ——— 153
- 4. 追跡…サアー, 追いかけてよう! ——— 164
- 5. 完成…メッセージや音を付ける ——— 173

CHAPTER 6 ● スクロール・ゲーム

- 1. 重ね合わせ…もはや一般教養です ——— 182
- 2. 割り込み…ウェイトと重ね合わせ ——— 191
- 3. QRL…パターン・コントロール言語 ——— 214
- 4. スカイ・ブルーザー…Playing Game ——— 235



● APPENDIX

1. MF-ASM2…PC88 シリーズ用アセンブラ ——— 262
2. インストラクション表…いわゆる ——— 276
3. ツール…Game Programming Kits ——— 284
4. マシン語命令小辞典…御一読アレツと! ——— 298



本書のキャラクタの作り方

0. パターン・エディタのプログラムを打ち込みセーブしておく
(Appendix 3 のパターン・エディタ「pated」)
1. 「pated」を実行する
2. キャラクタ・サイズの入力
3. キャラクタ・パターンの作製(パターン・エディタの使い方は、
2.3章参照)
4. パターンの作製が終了した時点で、パターン・エディタのEコマ
ンドを実行する
5. 次にデータ・タイプを入力、不必要なバンクを削除
6. パターン・データのセーブ(例 &HB500~B5BF にデータが作
製されている場合)

ディスクの場合

BSAVE "ファイル名", &HB500, &HB5BF-&HB500+1

テープの場合

MON 

h] W ファイル名, B500, B5BF

・パターン・データの転送

(例 &HB500~B5BF から &HB600~B6BF に転送)

MON 

h] MB500, B5BF, B600 

・すでにセーブしてあるデータとアペンドする場合

ディスクの場合

- ①アペンドするデータをロードしておく

BLOAD "ファイル名"

- ②データを未使用のメモリ・エリアに転送する
- ③必要なキャラクタ数だけ①と②を繰り返す この時転送する
メモリ・アドレスが連続していること 連続していないとセ
ーブができない
- ④データのセーブ……パターン・データのセーブ参照

テープの場合

- ①アペンドするデータをロードしておく

h] R ファイル名

- ②~③はディスクと同様
- ④パターン・データのセーブ参照

プレイ・ザ・ゲーム

(本書に掲載されているゲームの各場面)

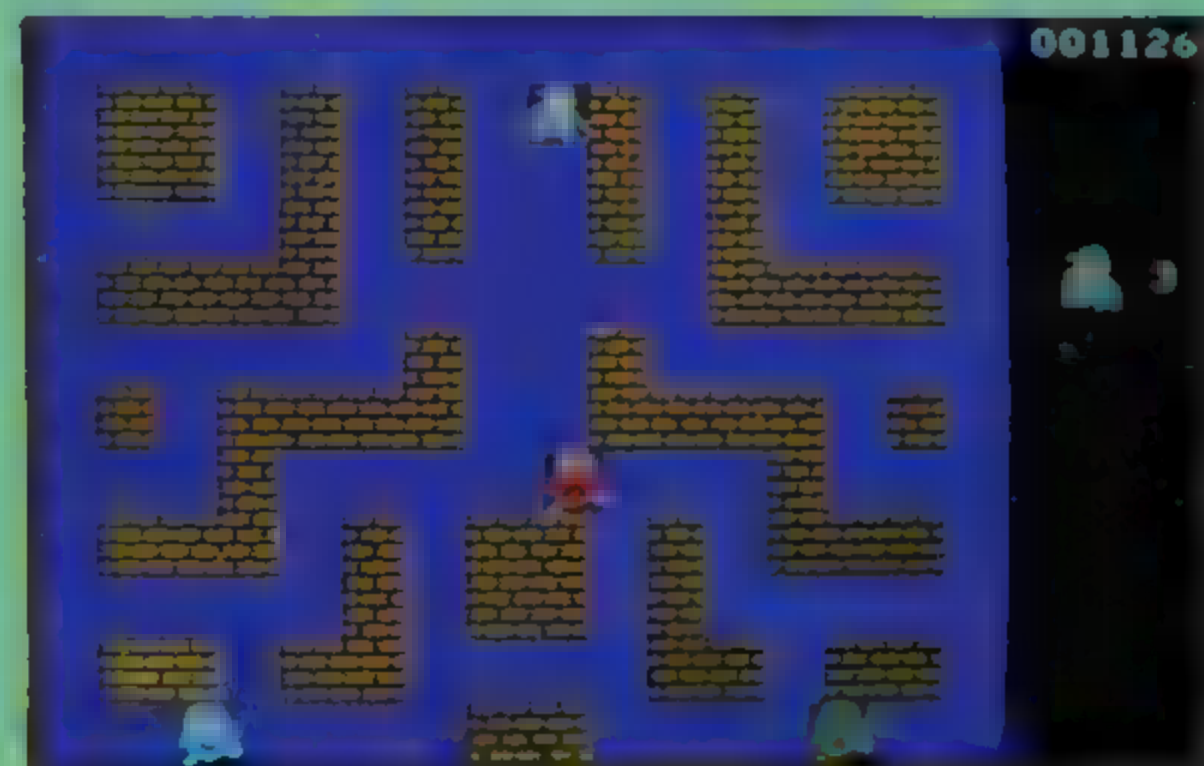
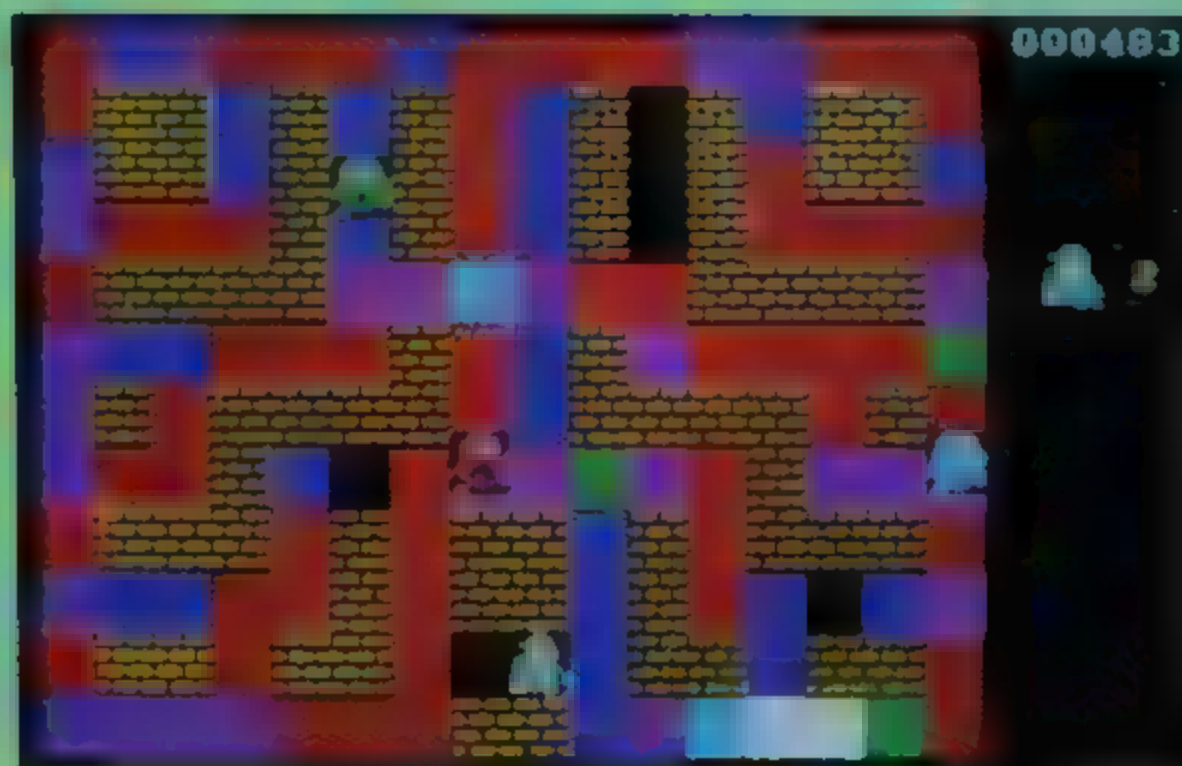
● スカイ・ブルーザー (6章より)



● シューティング・ゲーム (2章、3章より)



ペンキ・ボーイ (4章、5章より)



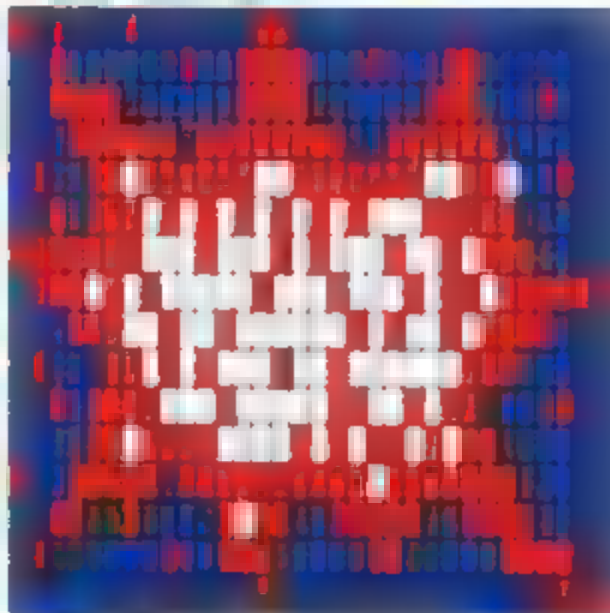
スカイ・ブルーザー (6章スクロールゲーム参照)

西歴ADC08年 思考のシステム化、感覚の絶対数値化理論により世界最大のネットワークマフィアとなったTH社。このTH社のおかげで、常にNo.2でしかないGI社。このGI社の常務であるあなたは、No.1になるためには世論を味方にするのが一番と判断した。そこで、思考システムや数値化は、人間の尊厳を奪うものであるという運動を推進した。これがもとで、ついに会社間戦争となる。

GI社常務のあなたは、旧式だがセミオート照準を持つクテノボマIIIを操り、TH社の奥深くにあるストラクチャー・ボールを破壊することにした。このストラクチャー・ボールこそTH社を運営しているエキスパート・システムなのだ。

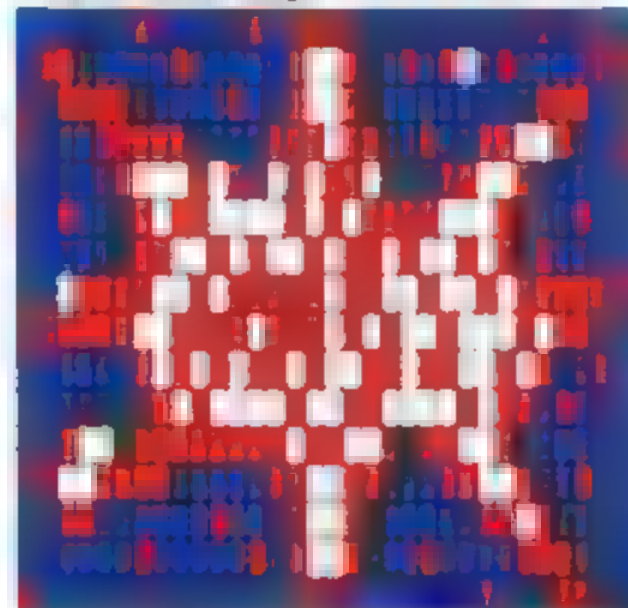
●スクロール・ゲーム用キャラクタ…地上パターン(6章参照) ■注: 20, 22, 24, 21, 23, 25は同じパターン

* 20 22 24



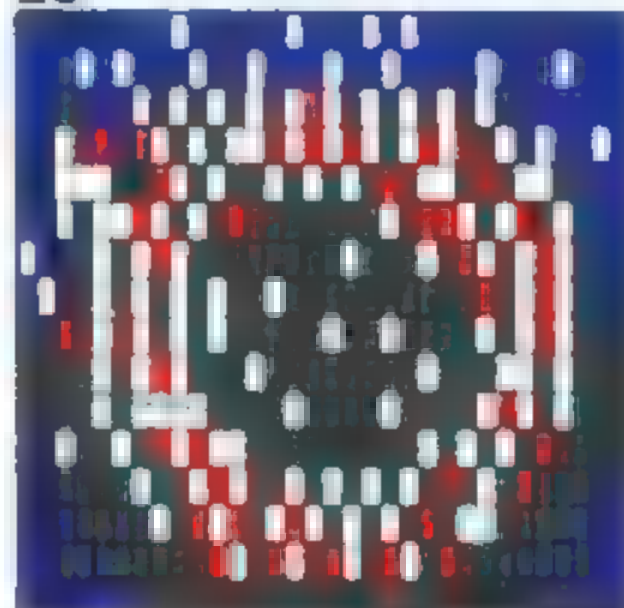
セーブ・アドレス 6E80 ~ 6EFFH

* 21 23 25



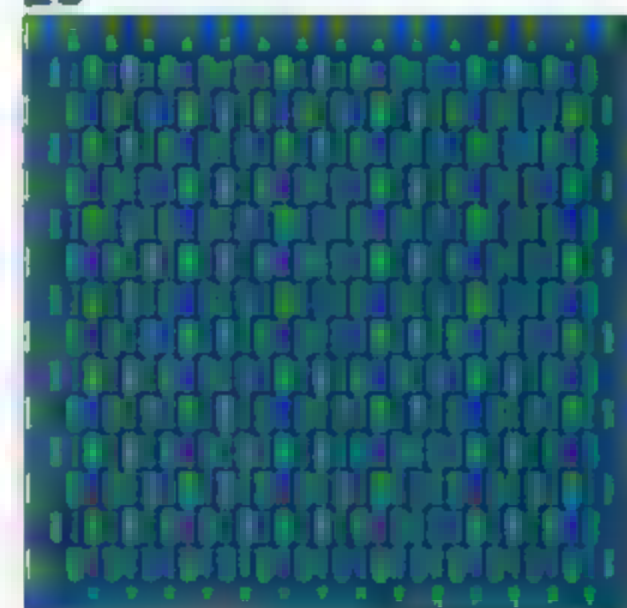
6F00 ~ 6F7FH

26



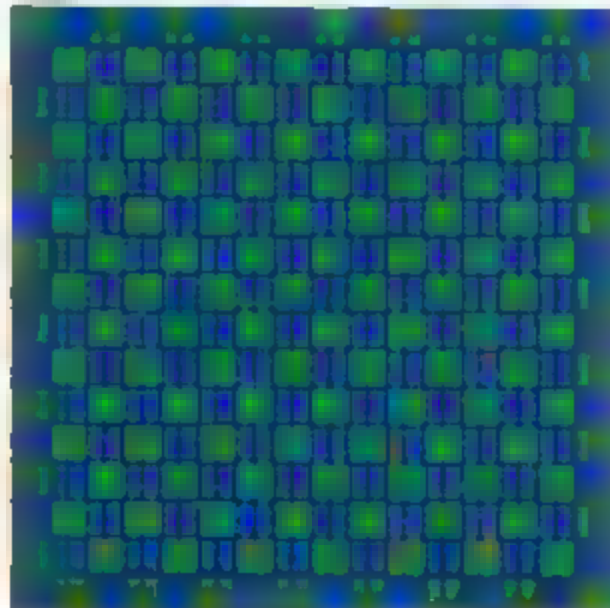
6F80 ~ 6FFFH

28



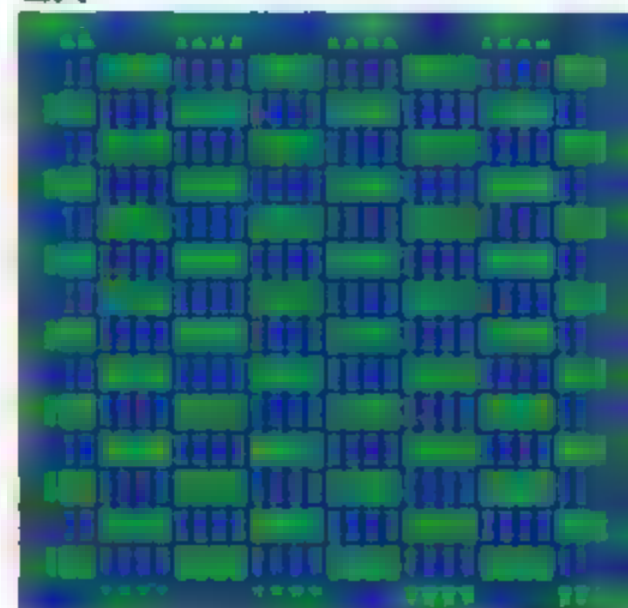
7000 ~ 703FH

29



7040 ~ 707FH

2A



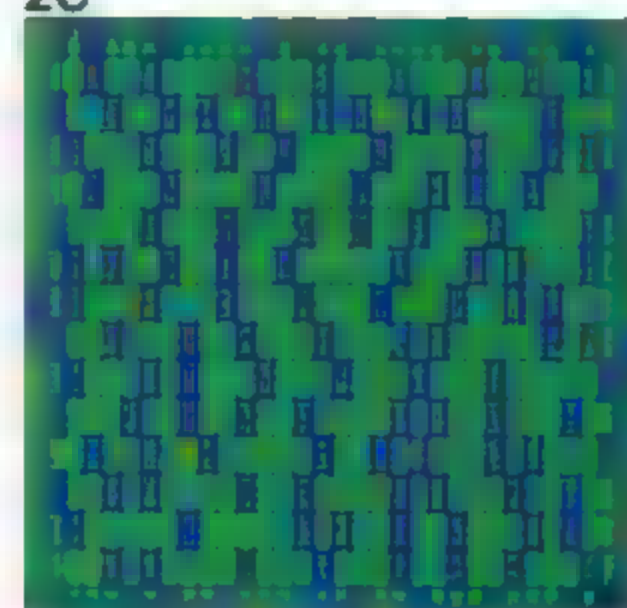
7080 ~ 70BFH

2B



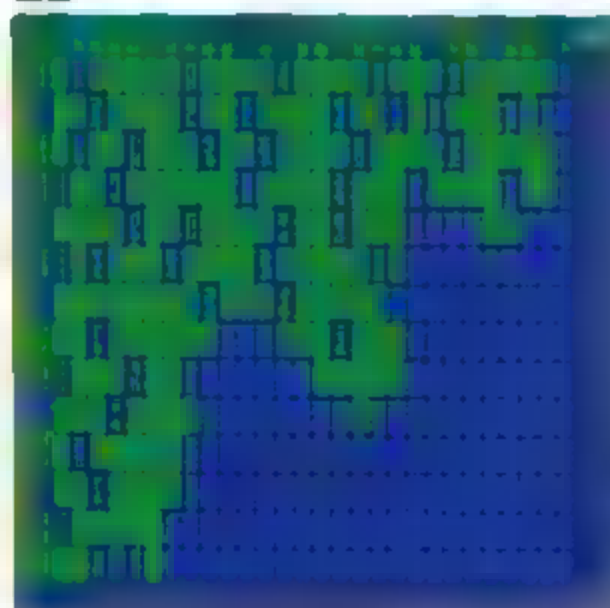
70C0 ~ 70FFH

2C



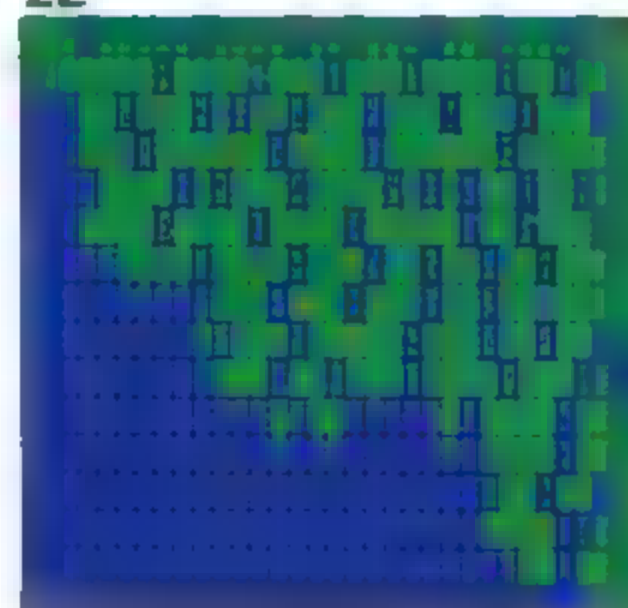
7100 ~ 713FH

2D



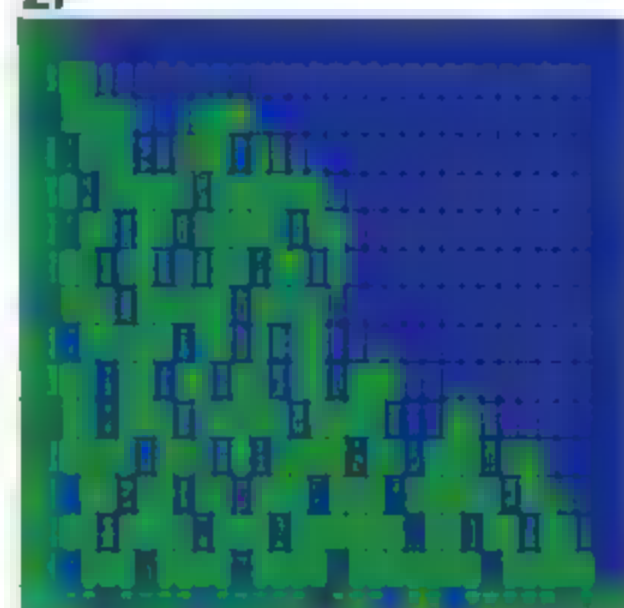
7140 ~ 717FH

2E



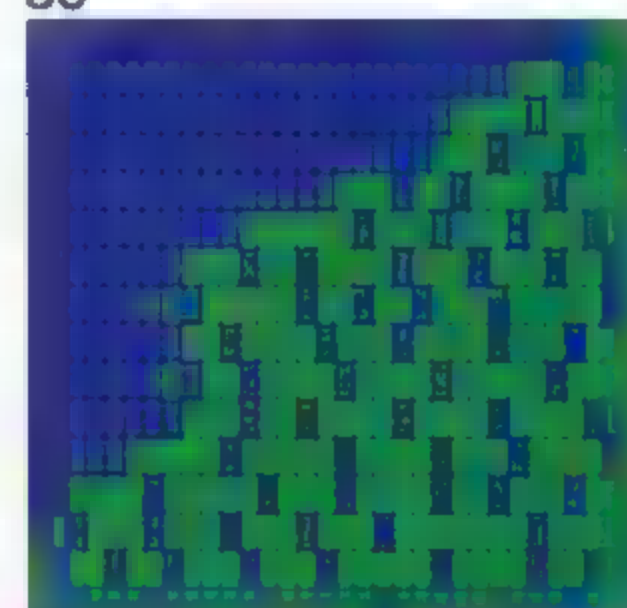
7180 ~ 71BFH

2F



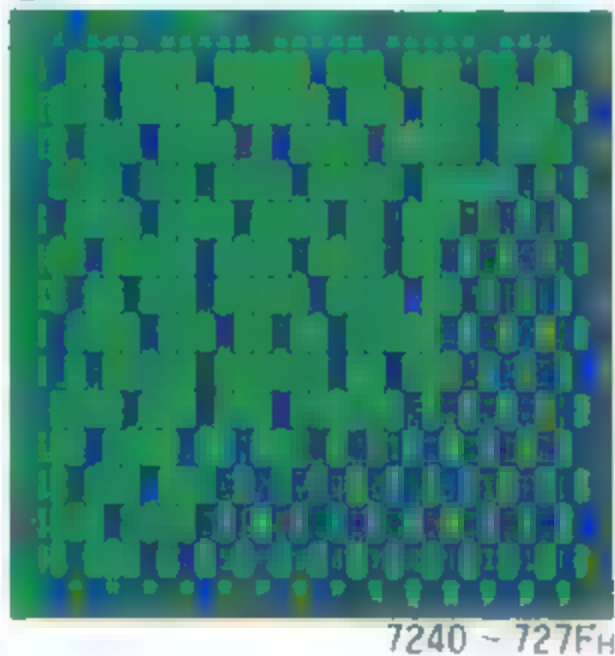
71C0 ~ 71FFH

30



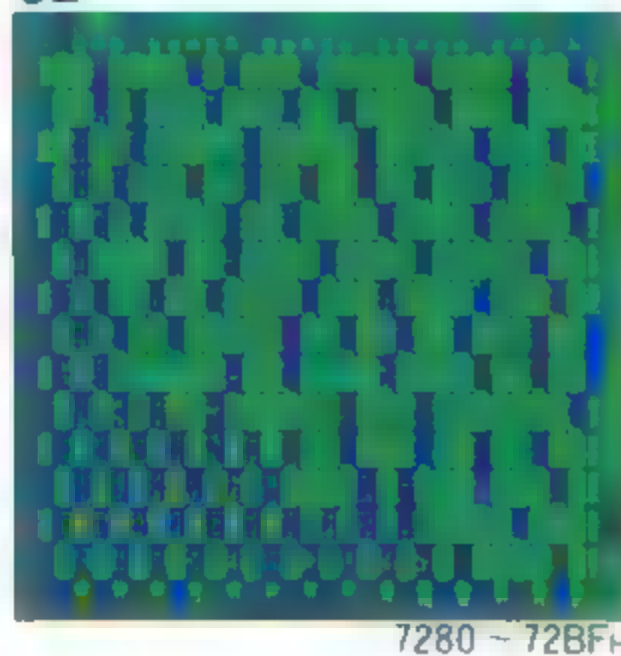
7200 ~ 723FH

31



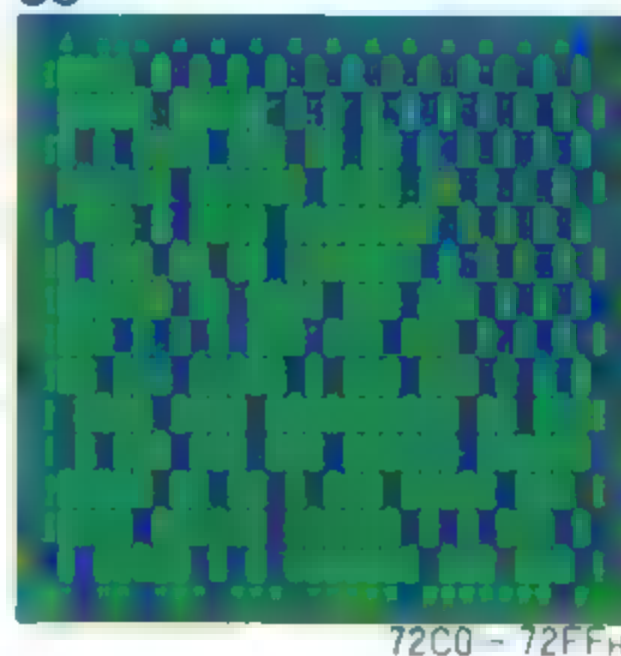
7240 ~ 727FH

32



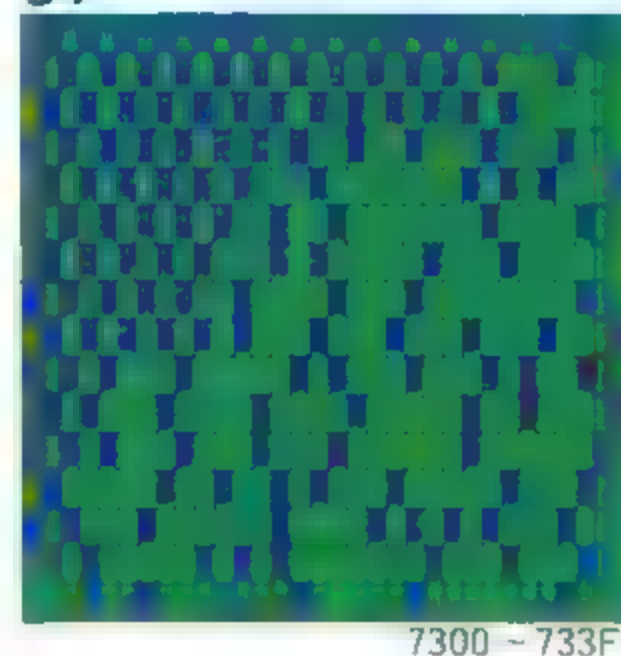
7280 ~ 72BFH

33



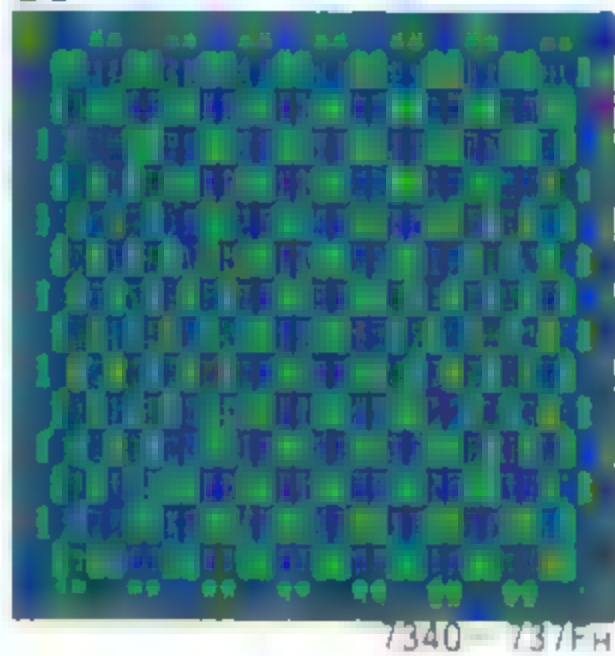
72C0 ~ 72FFH

34



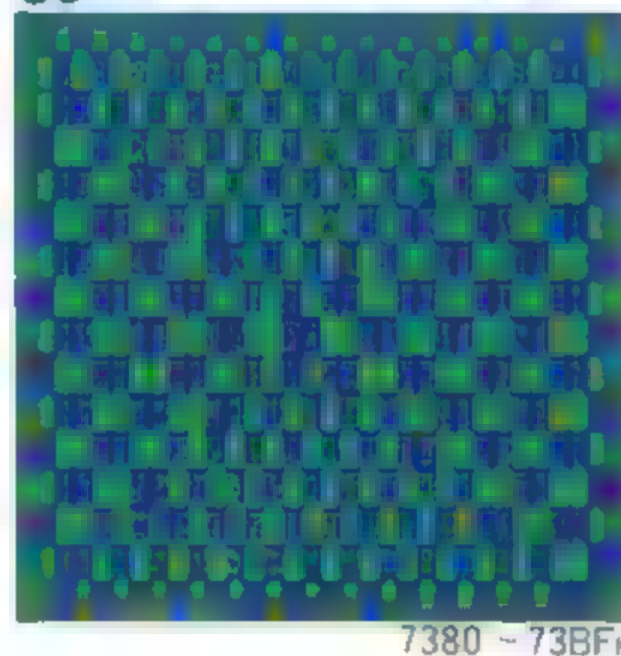
7300 ~ 733FH

35



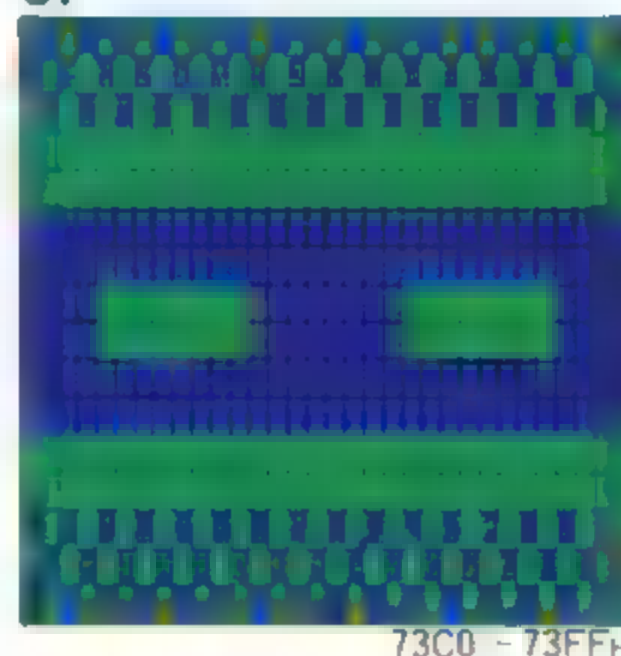
7340 ~ 737FH

36



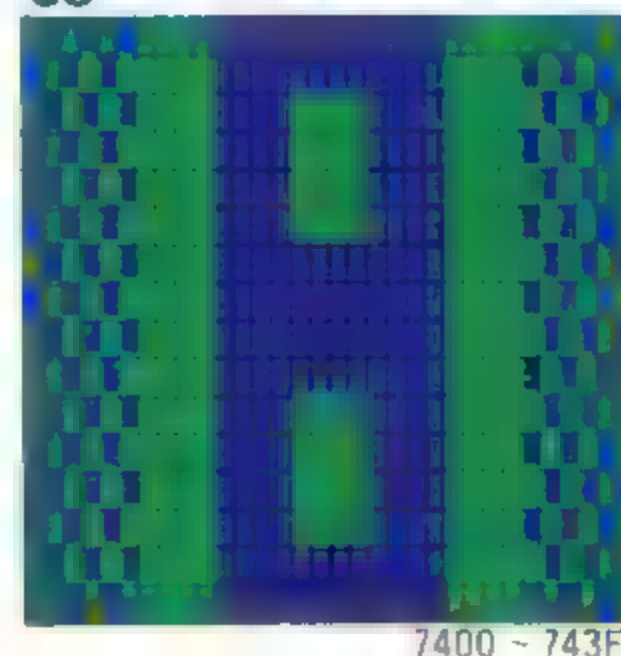
7380 ~ 73BFH

37



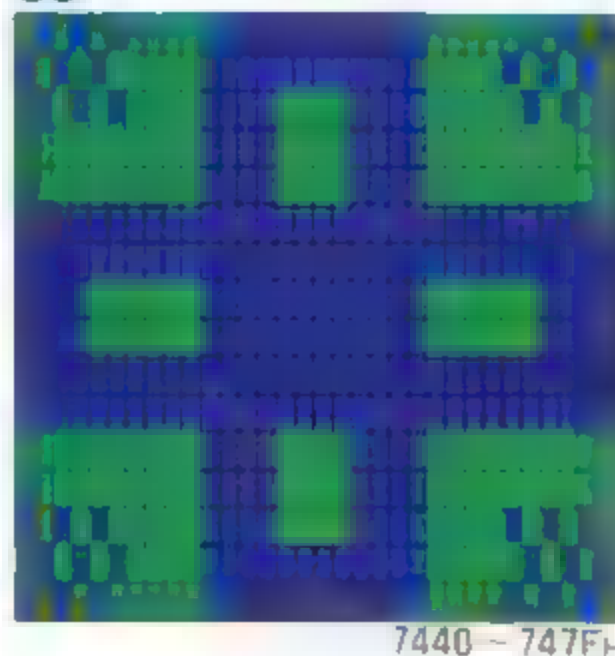
73C0 ~ 73FFH

38



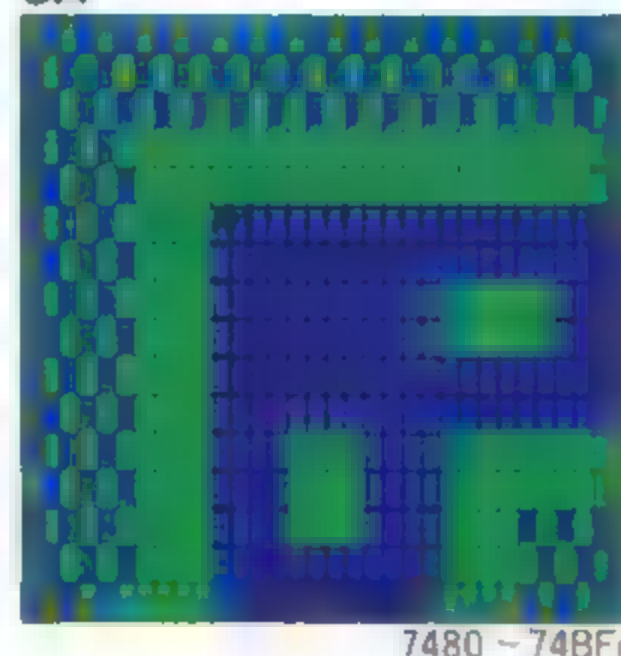
7400 ~ 743FH

39



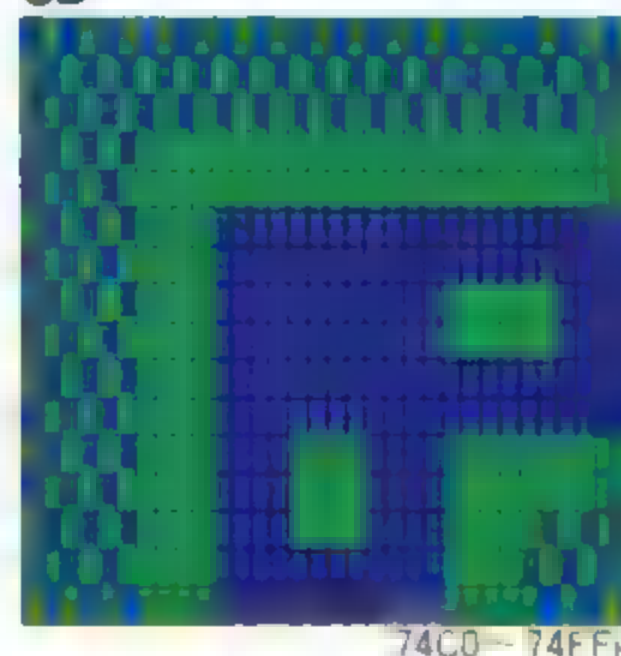
7440 ~ 747FH

3A



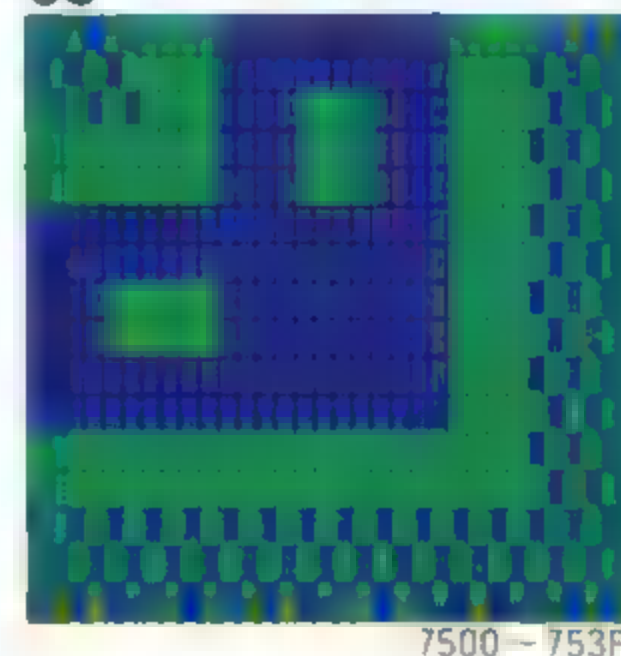
7480 ~ 74BFH

3B



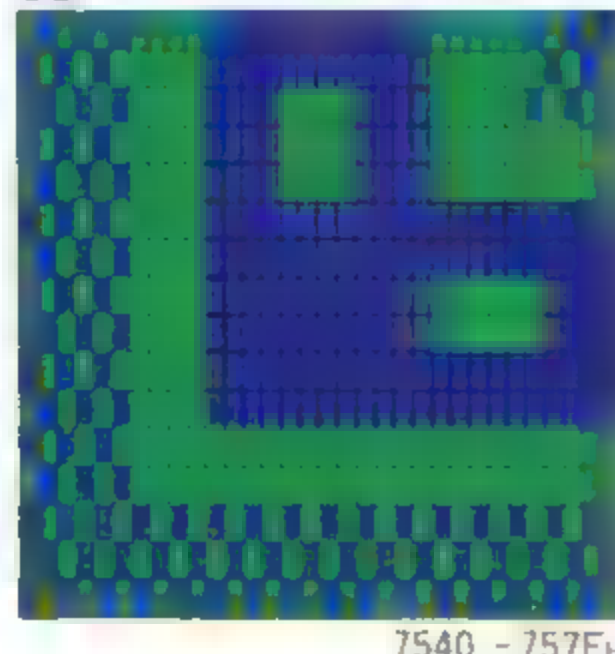
74C0 ~ 74FFH

3C



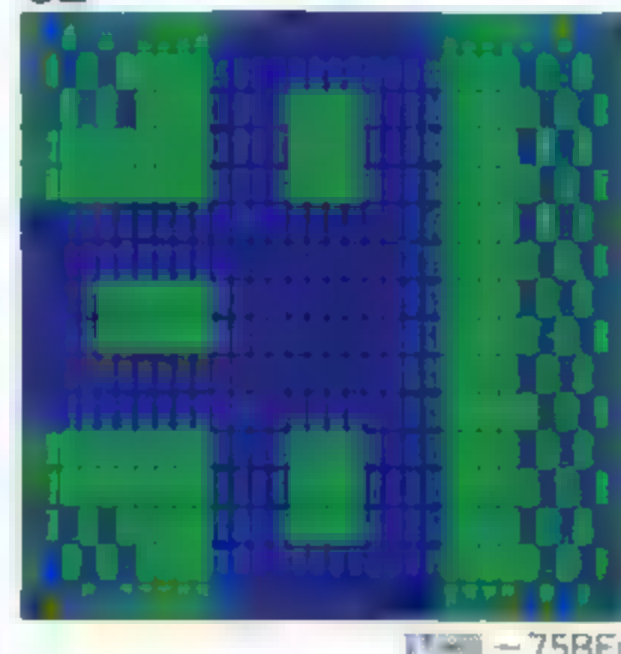
7500 ~ 753FH

3D



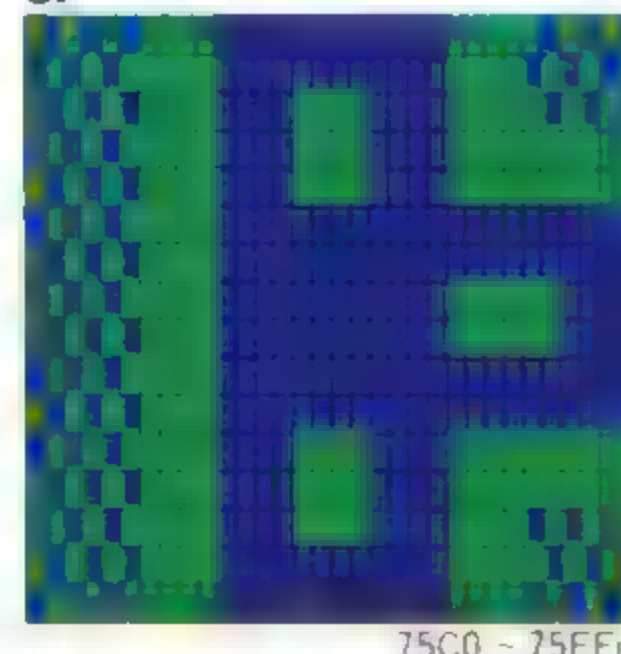
7540 ~ 757FH

3E



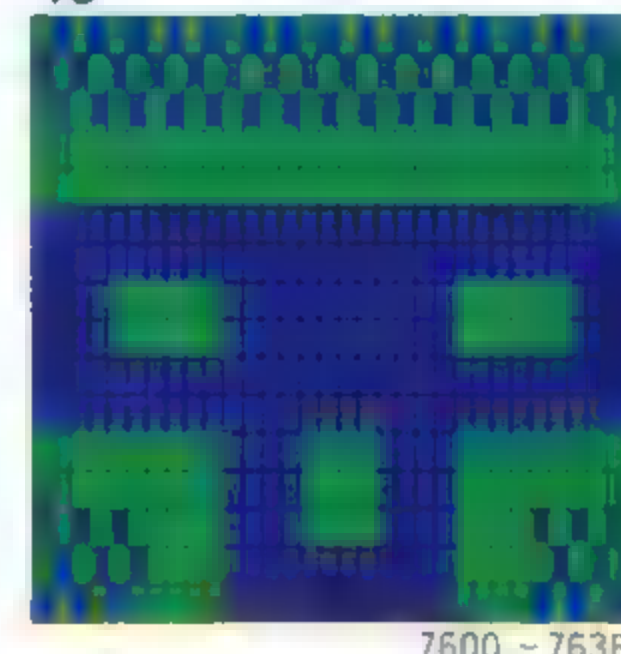
7580 ~ 75BFH

3F



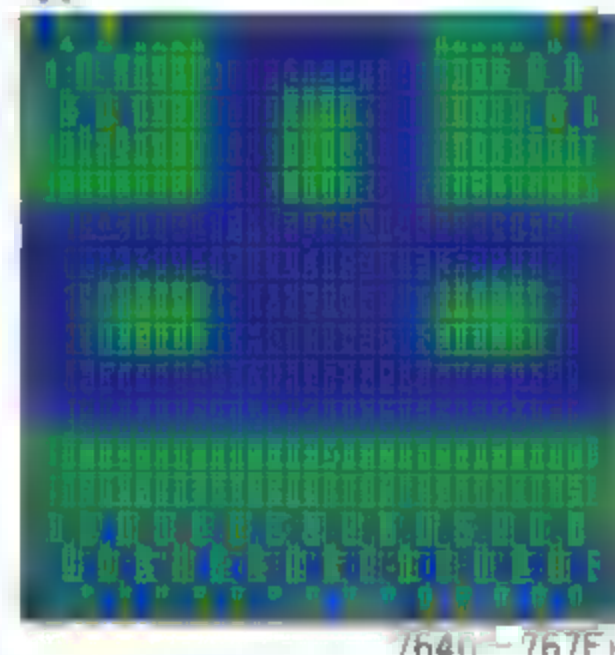
75C0 ~ 75FFH

40



7600 ~ 763FH

41



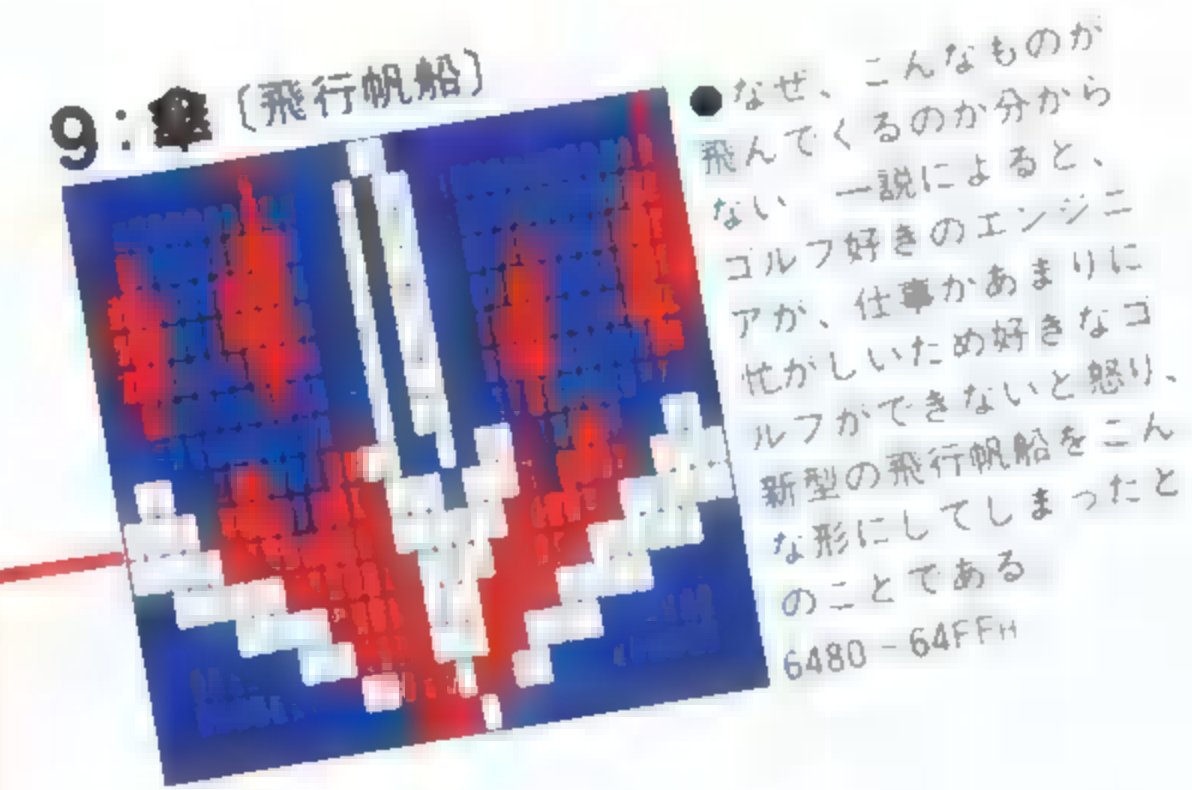
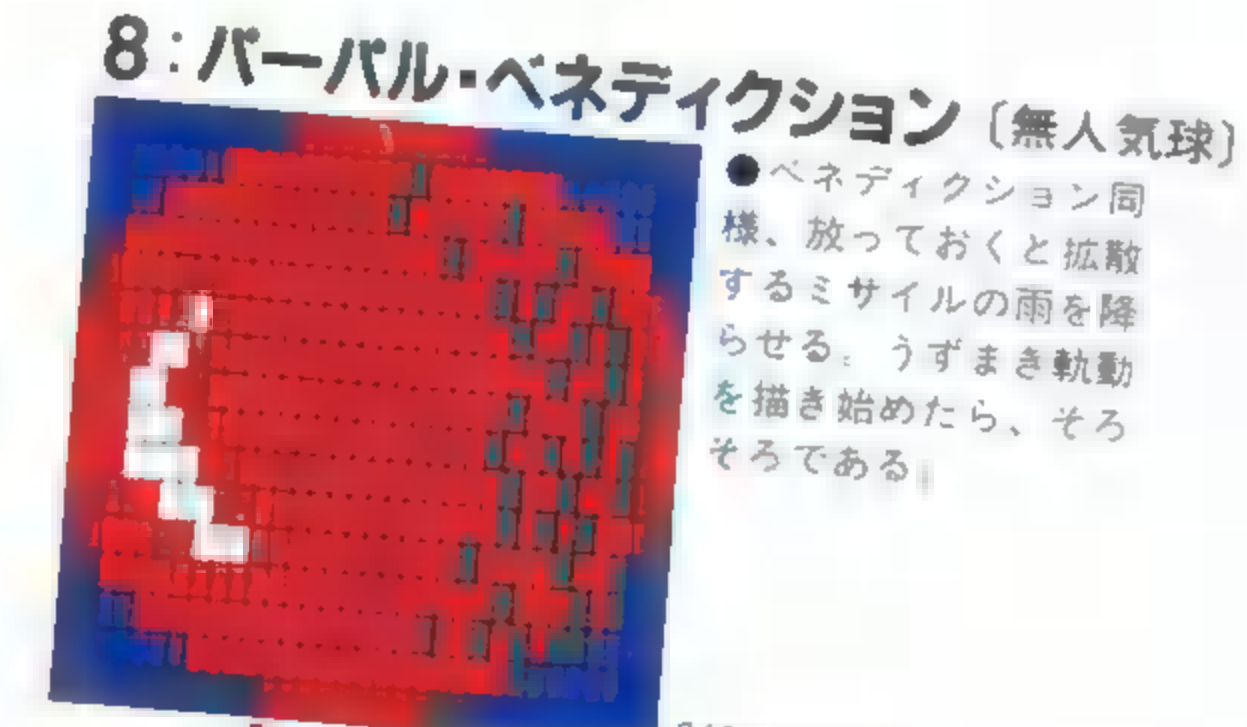
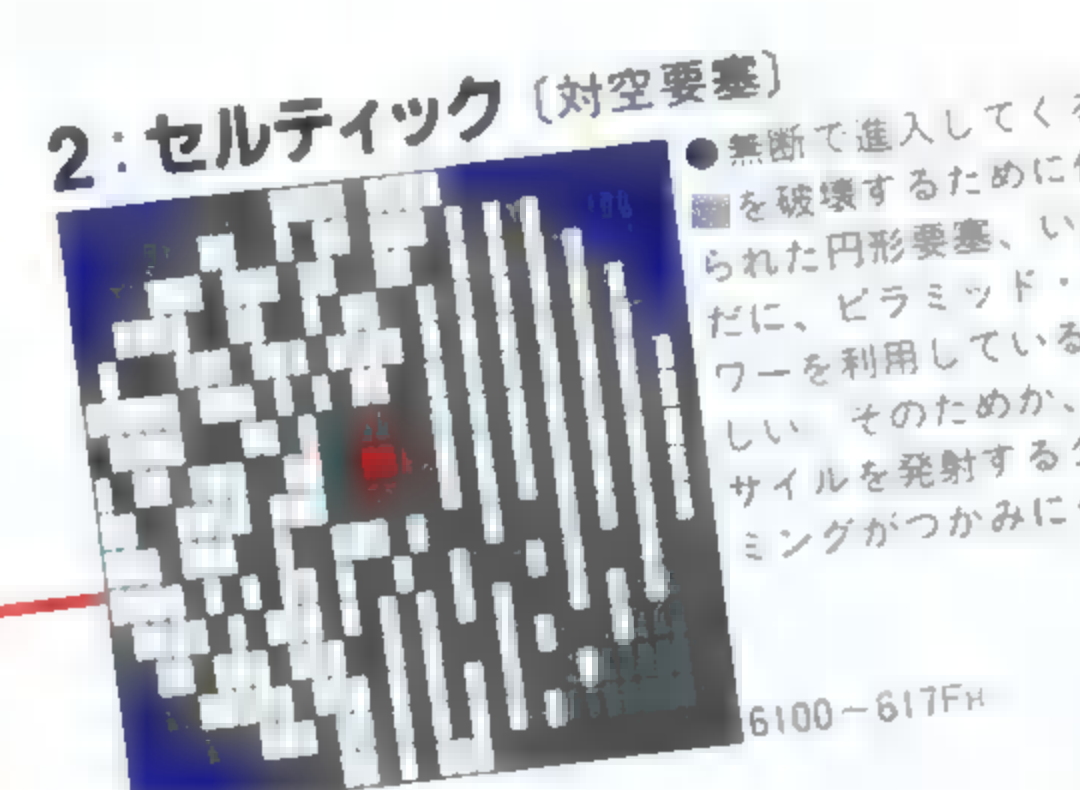
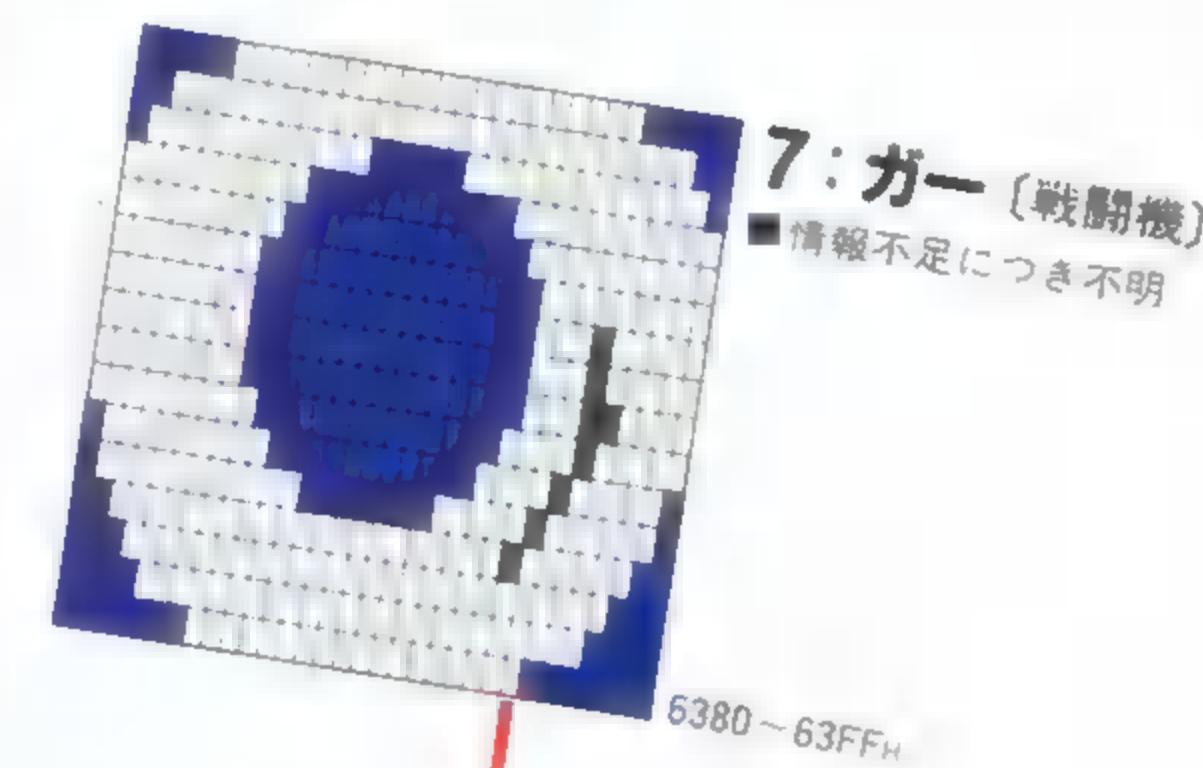
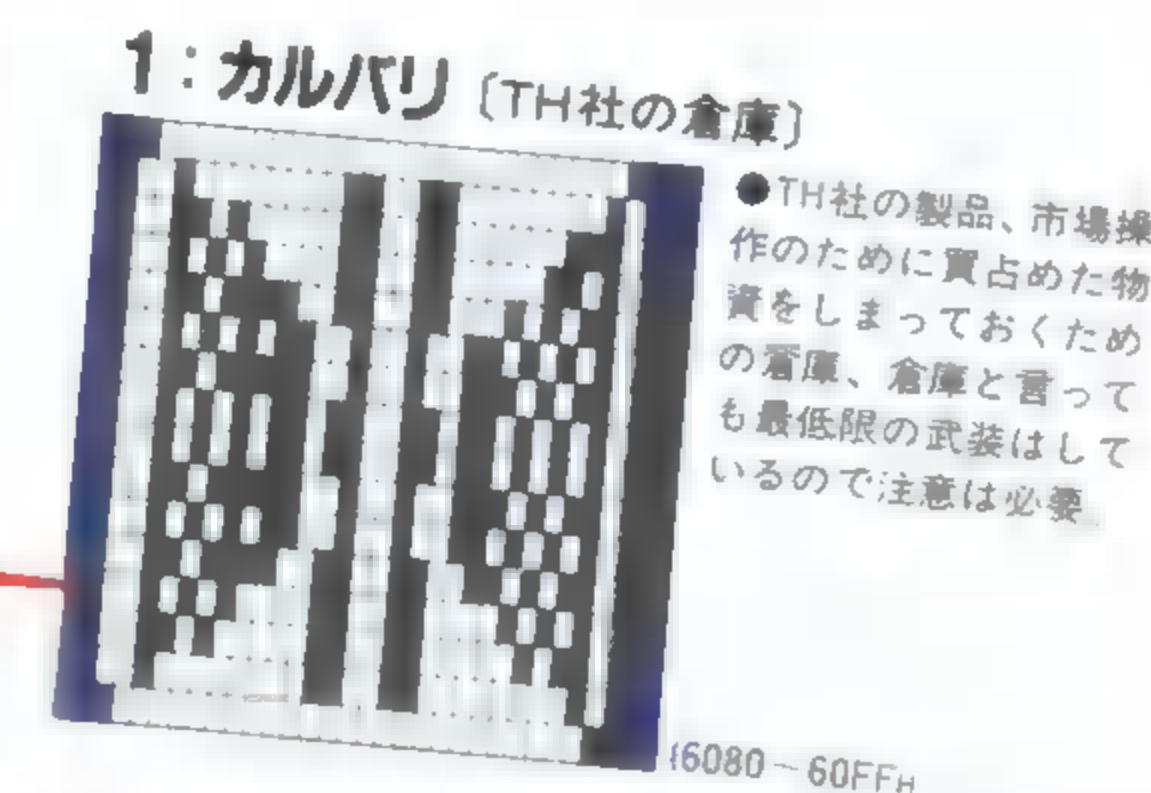
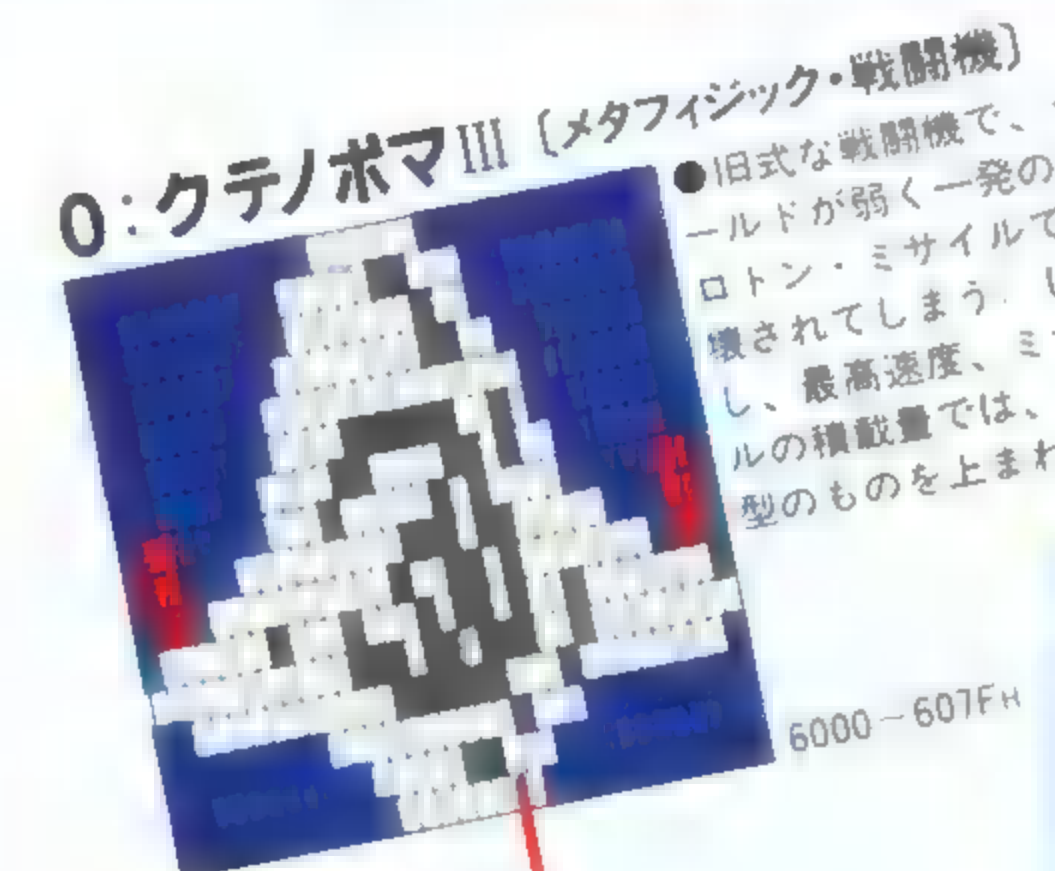
7640 ~ 767FH



7F00 ~ 7F07H

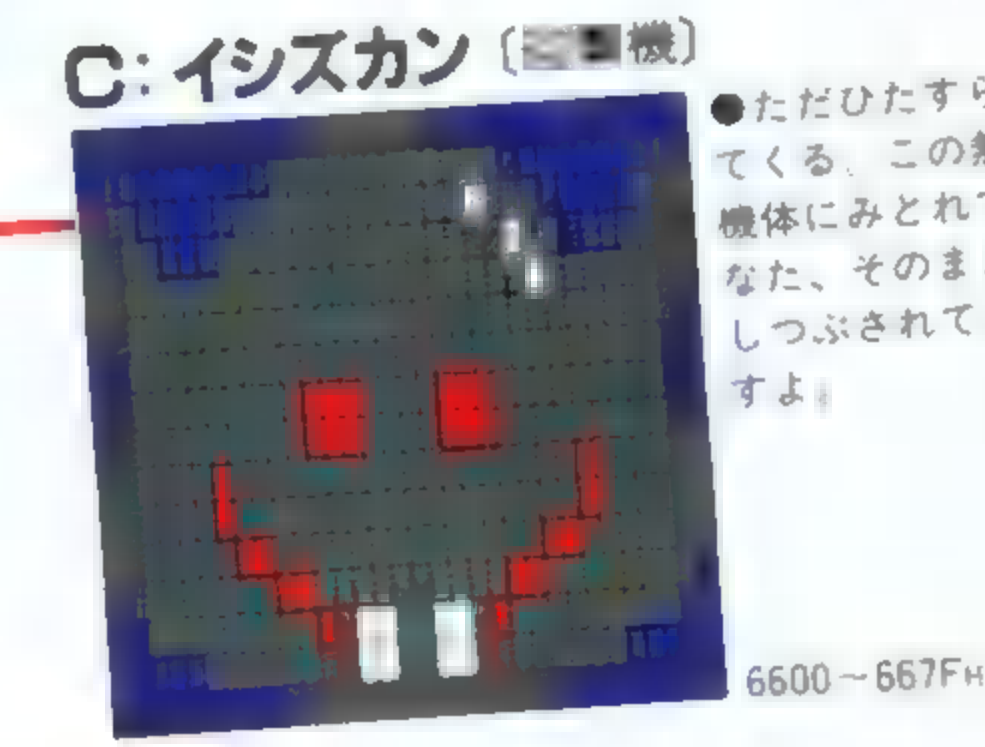
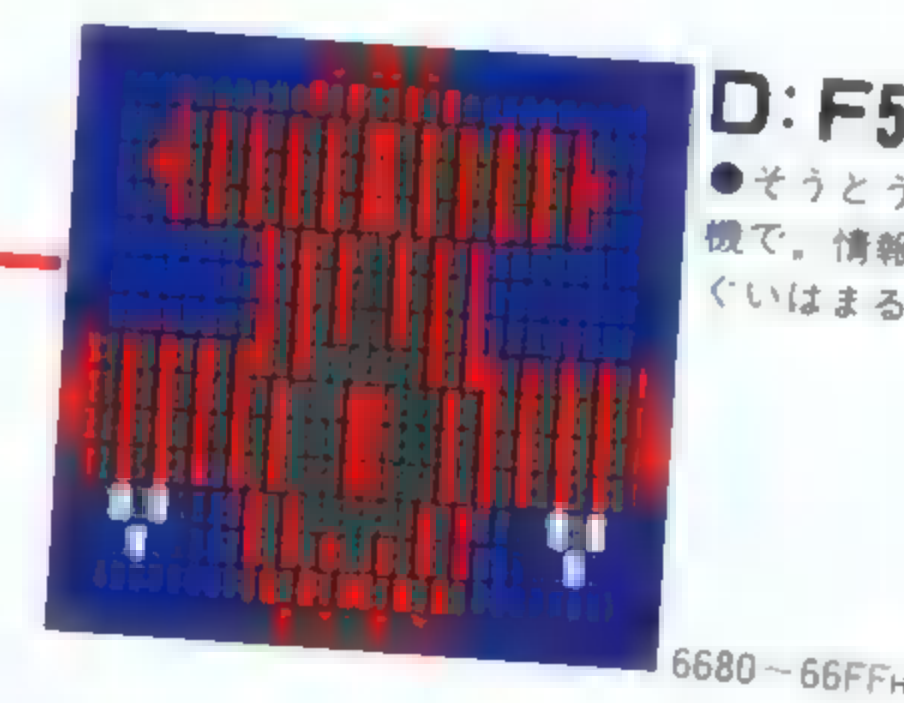
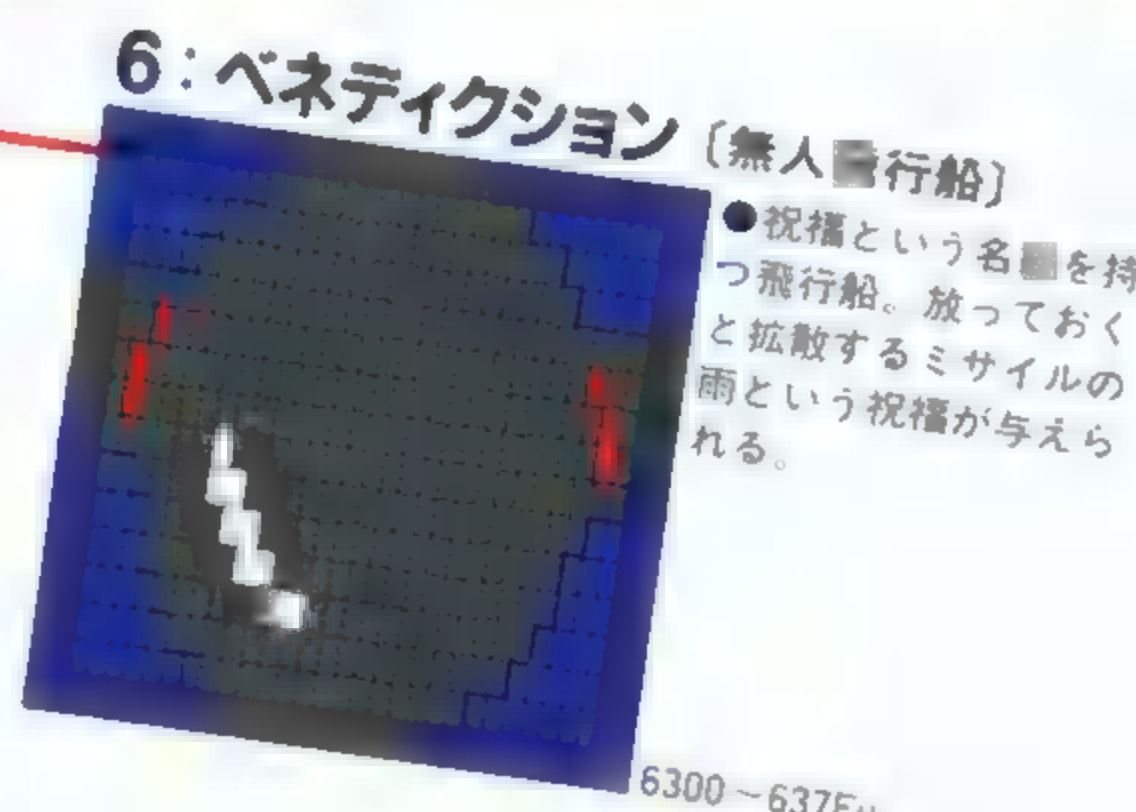
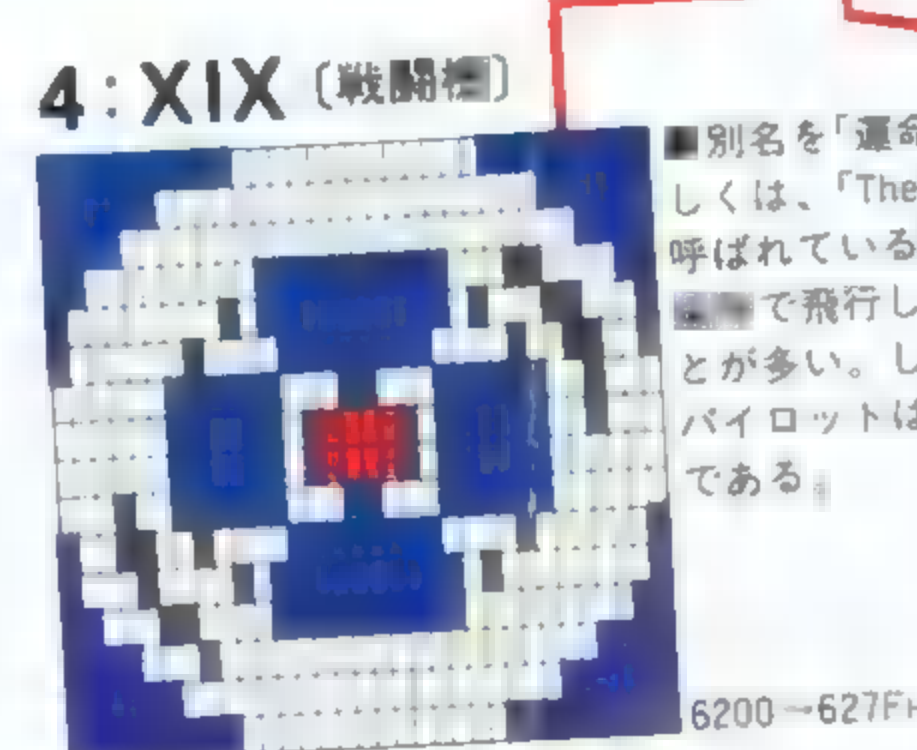
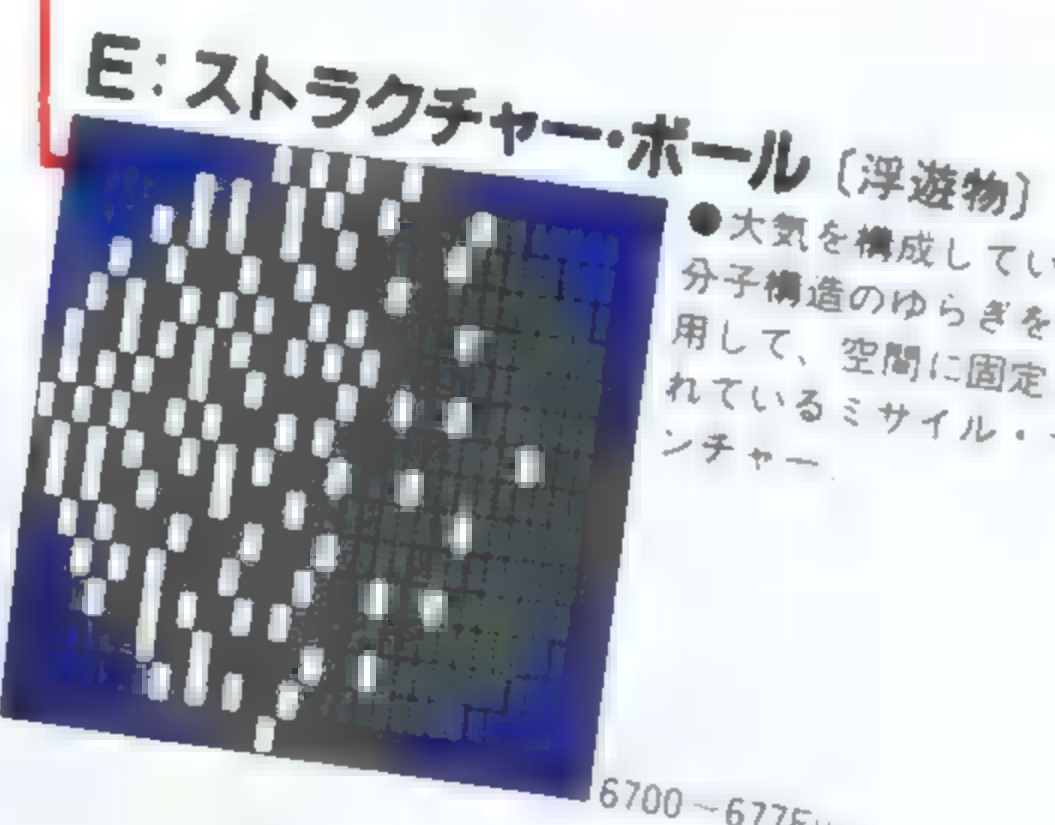
パレットコード	色	パレットコード	色
0	黒	5	シアン
1	青	6	黒
2	赤	7	白
3	マゼンタ	(8)	透明
4	緑		

スカイ・ブルーザー用パターンと格納アドレス



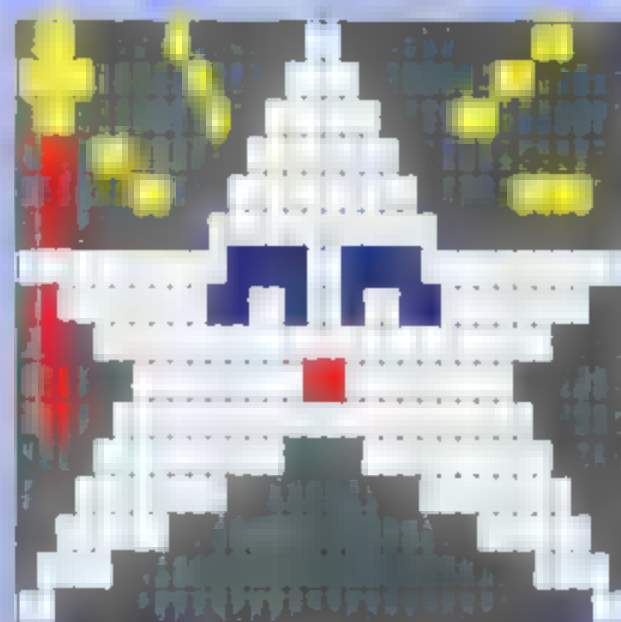
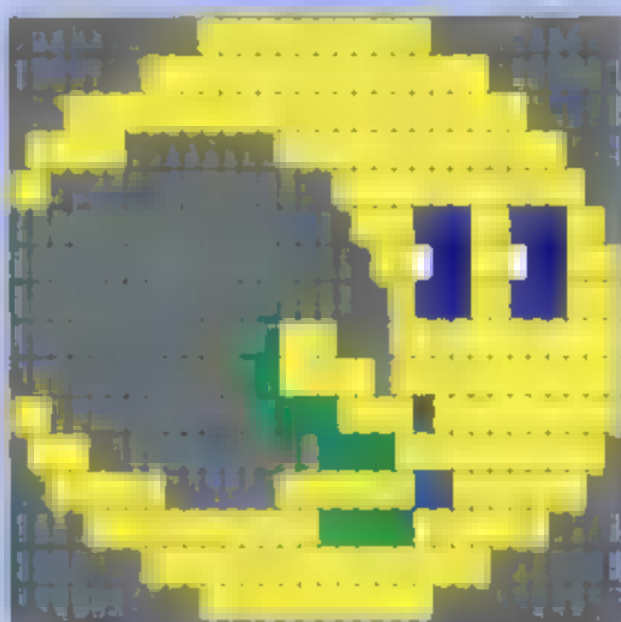
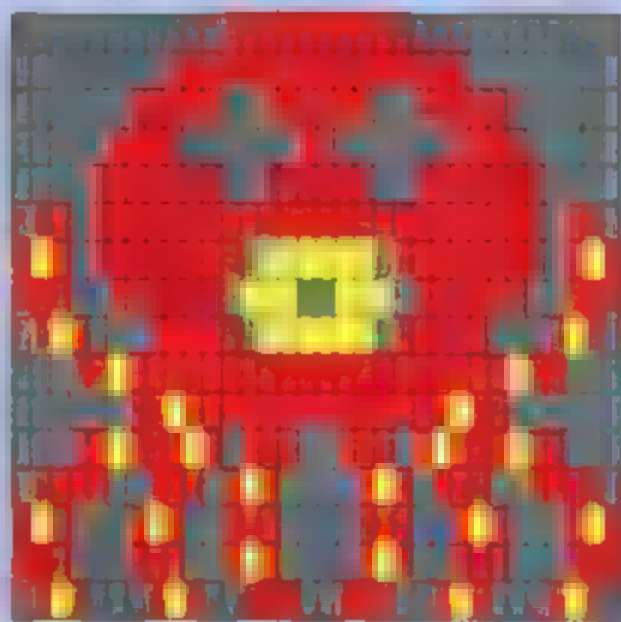
START

END



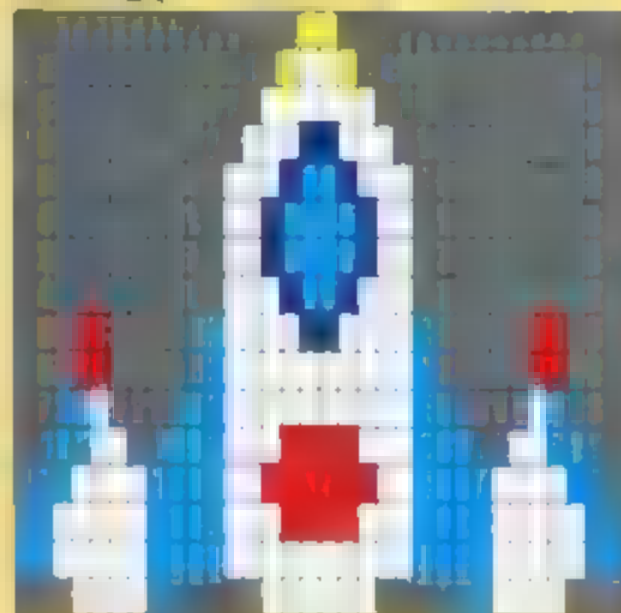
キャラクタ・パターン集

シューティング
・ゲームの
敵キャラクタ
(2,3章参照)



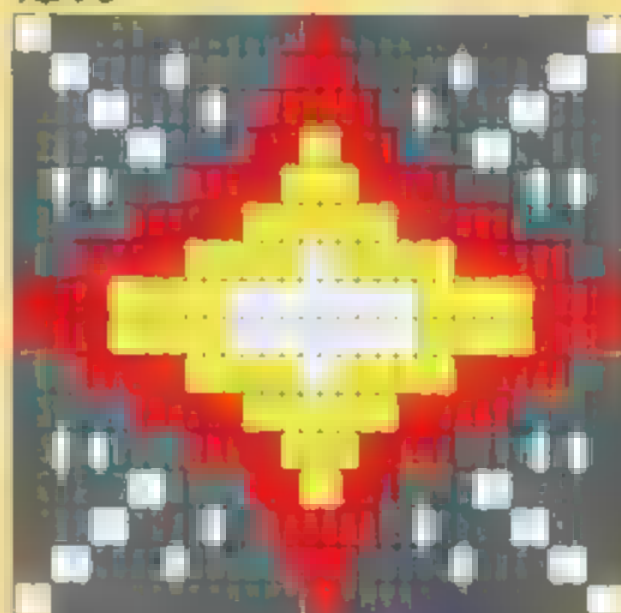
主人公と
弾のキャラクタ
(2,3章参照)

主人公



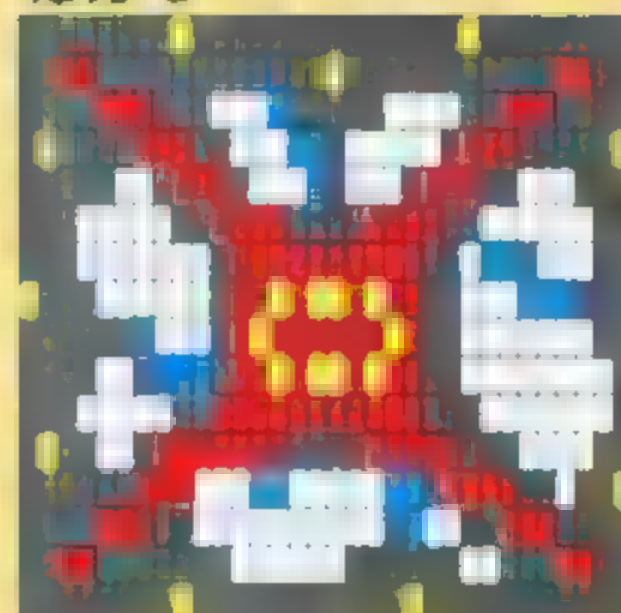
セーブ・アドレス B600H~B6BFH

爆発-1



B900H~B9BFH

爆発-2



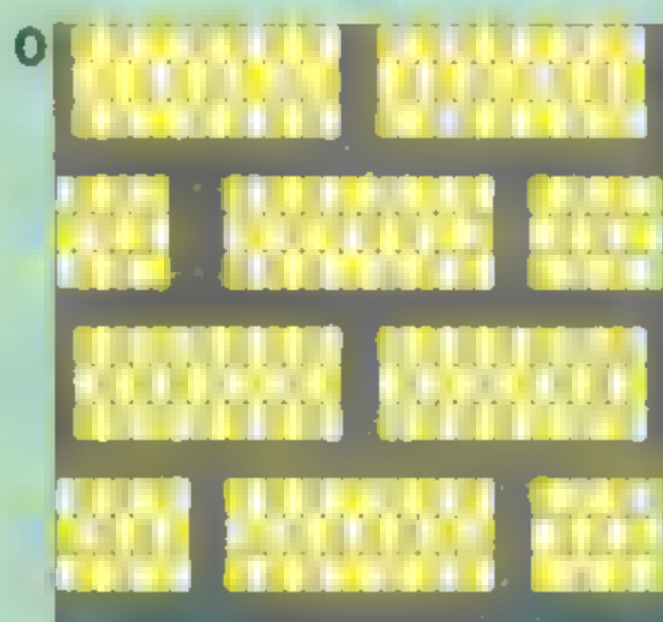
B9C0H~BA7FH

弾

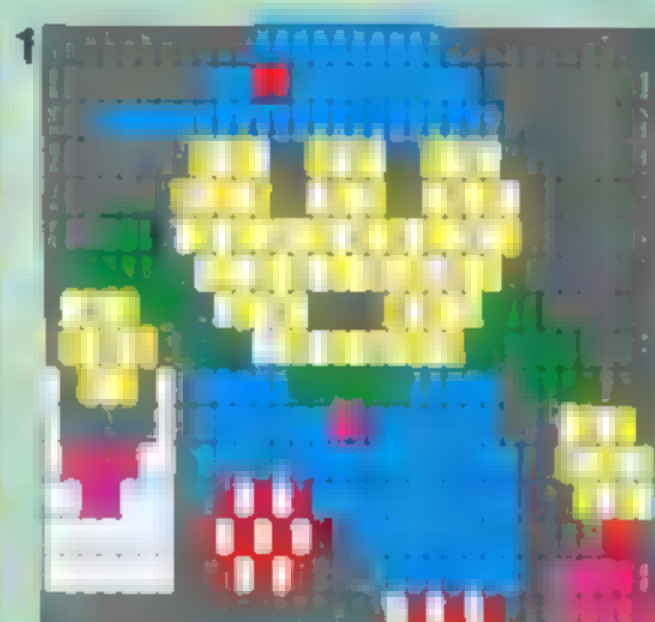


BA80H~BA8BH

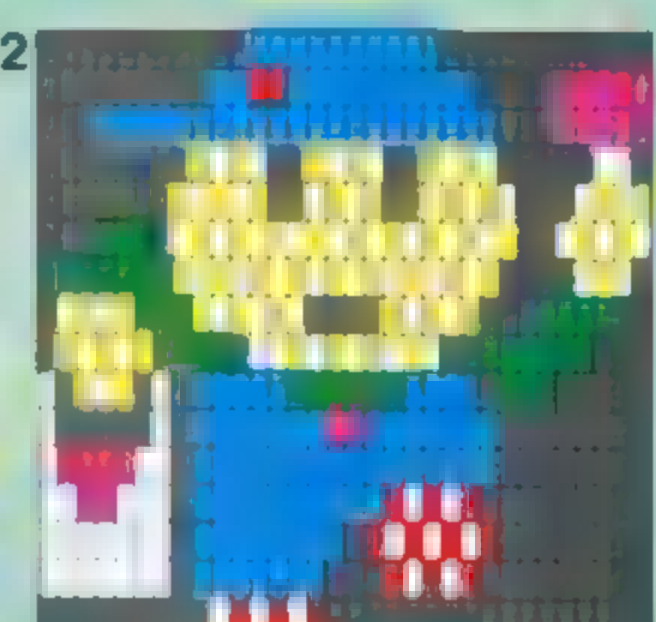
迷路型ゲームの
キャラクタ
(4,5章参照)



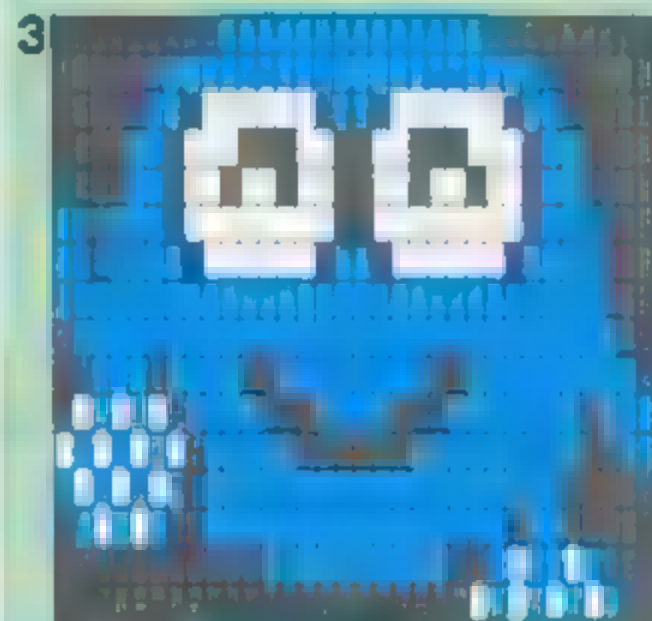
セーブ・アドレス B600H~B6BFH



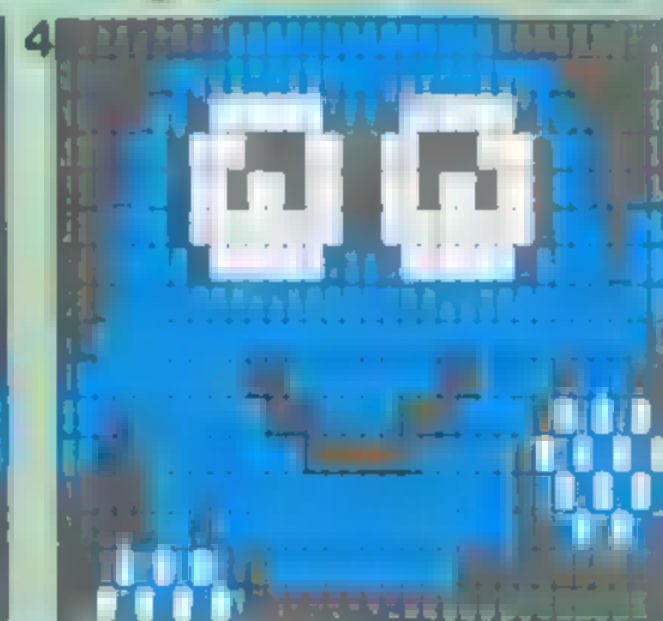
B6C0H~B67FH



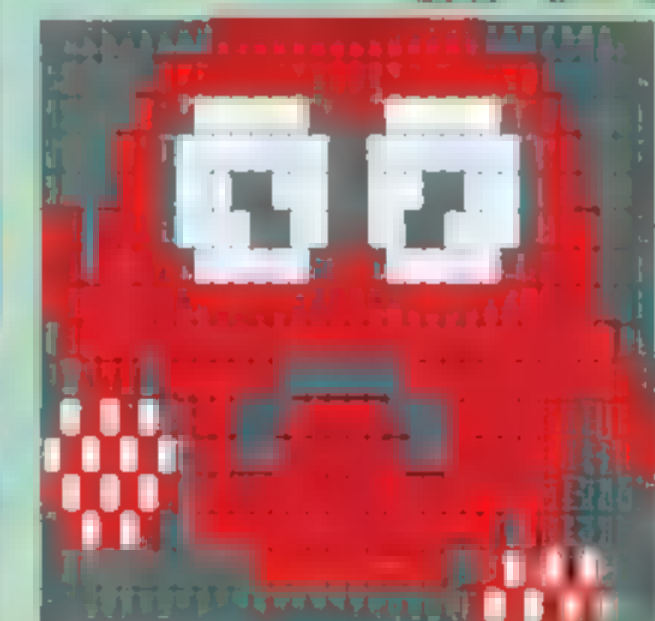
B780H~B83FH



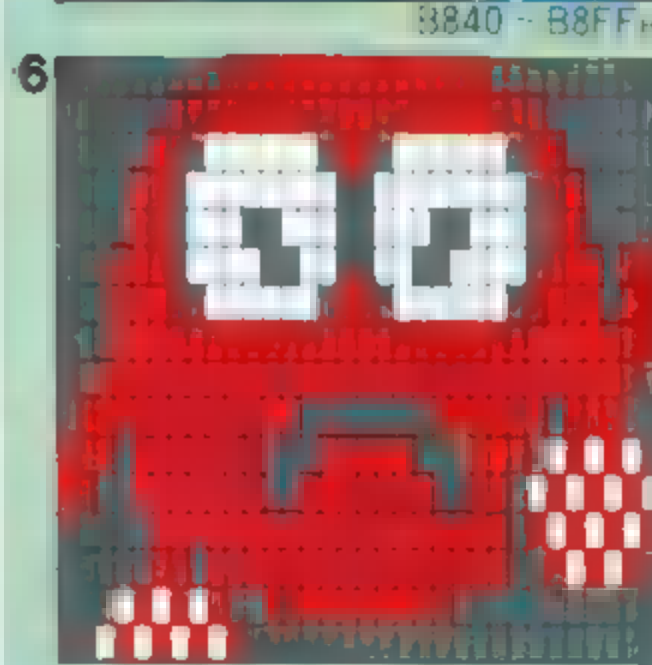
B840H~B8FFH



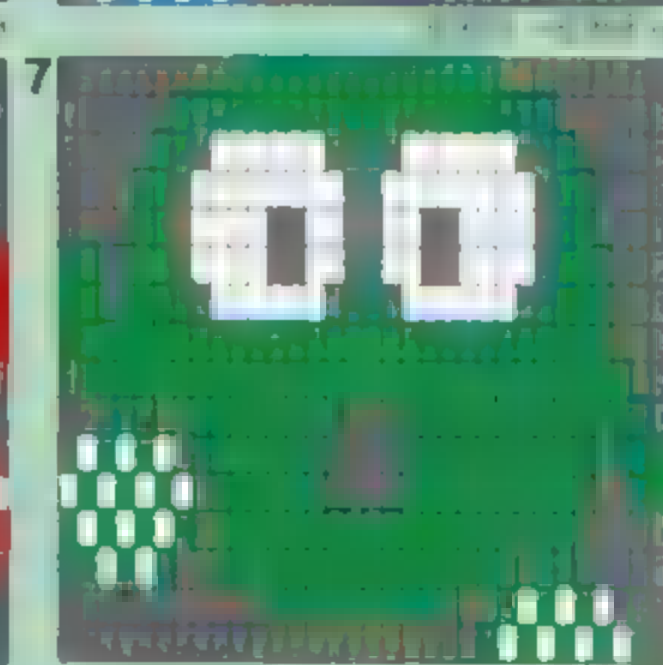
B840H~B8FFH



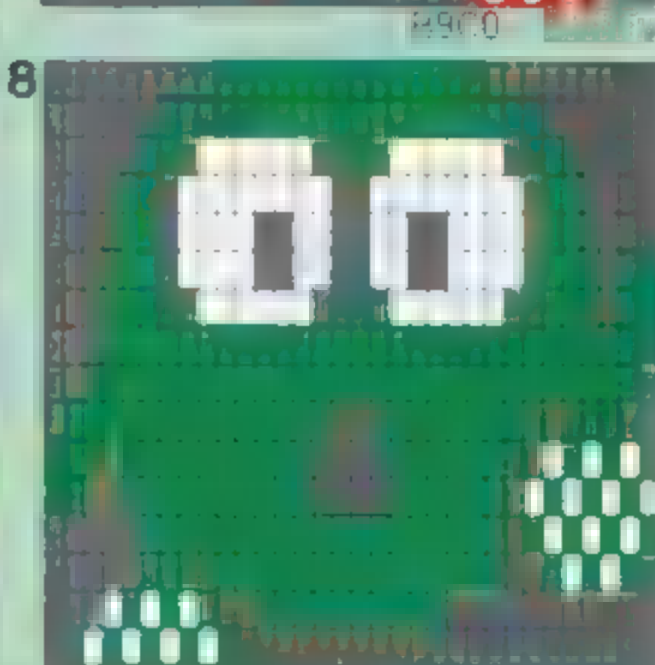
B9C0H



BA80H~BB3FH



BB40H~BBFFH



BC00H~BCBFH

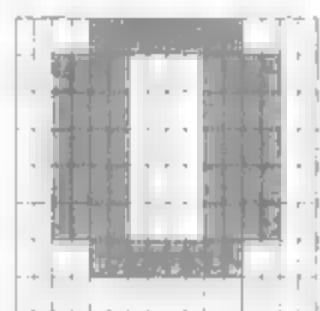
重ね合わせ用
テスト・パターン(6色)

移動パターン

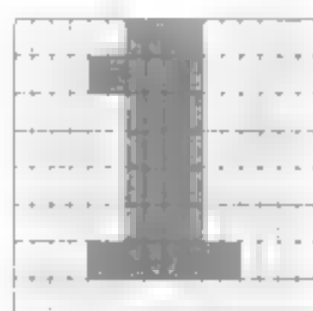


セーブ・アドレス B600H~B6FFH

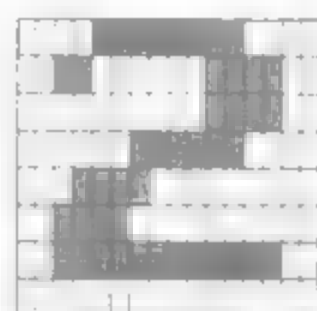
数字・文字パターンとデータ格納アドレス(3章参照)



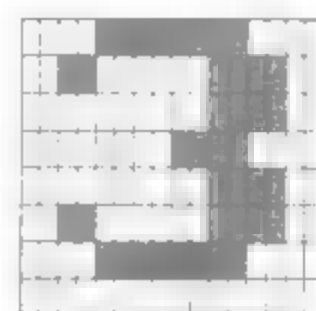
BB00H ~
BB0FH



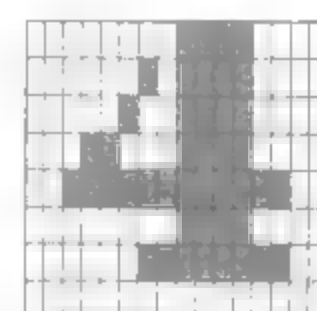
BB10H ~
BB1FH



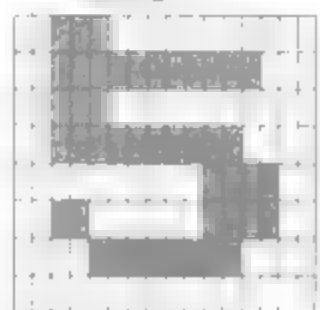
BB20H ~
BB2FH



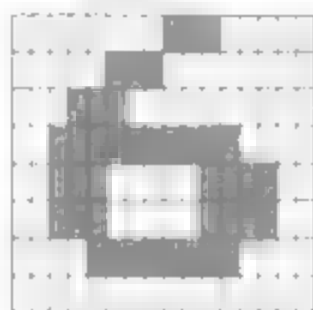
BB30H ~
BB3FH



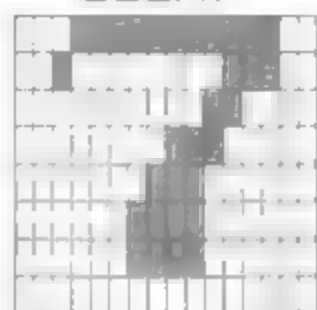
BB40H ~
BB4FH



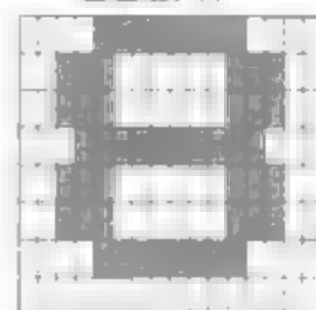
BB50H ~
BB5FH



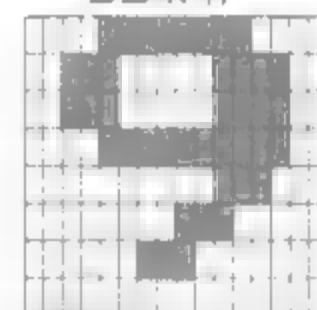
BB60H ~
BB6FH



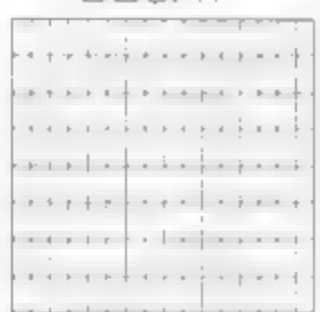
BB70H ~
BB7FH



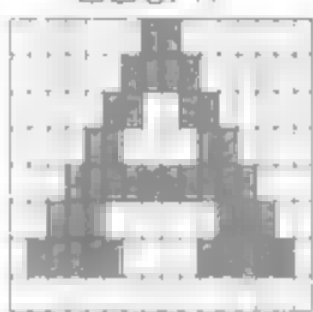
BB80H ~
BB8FH



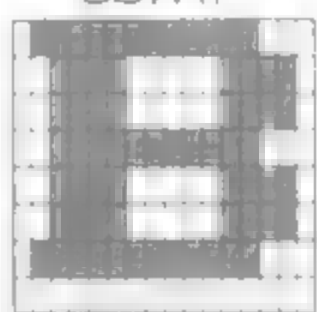
BB90H ~
BB9FH



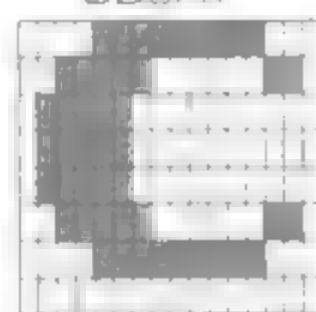
BBA0H ~
BBAFH



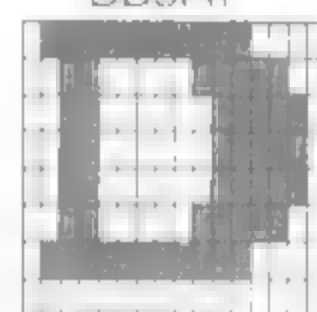
BBB0H ~
BBBFH



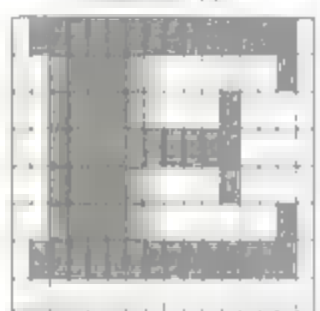
BBC0H ~
BBCFH



BBD0H ~
BBDFH



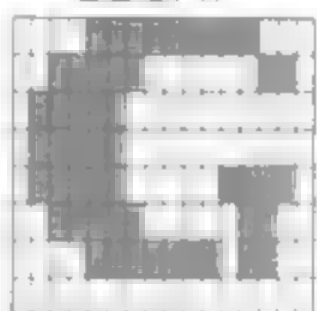
BBE0H ~
BBEFH



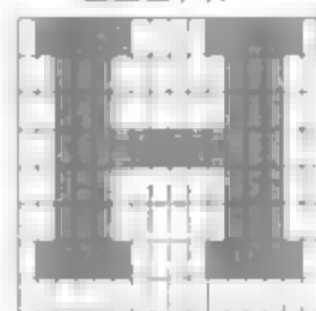
BBF0H ~
BBFFH



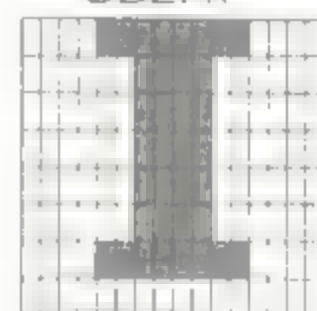
BC00H ~
BC0FH



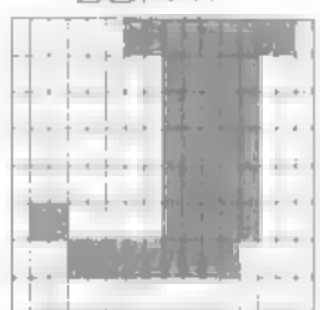
BC10H ~
BC1FH



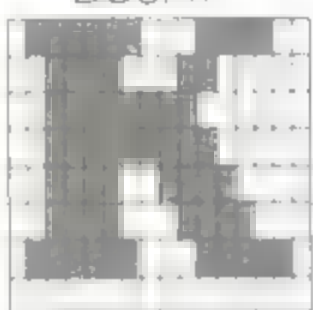
BC20H ~
BC2FH



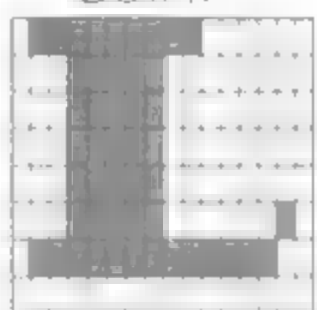
BC30H ~
BC3FH



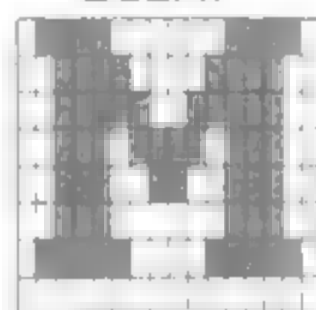
BC40H ~
BC4FH



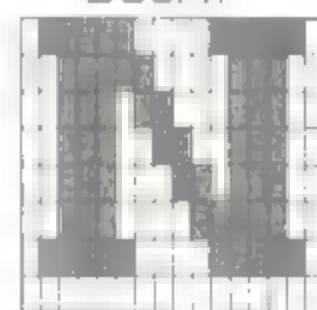
BC50H ~
BC5FH



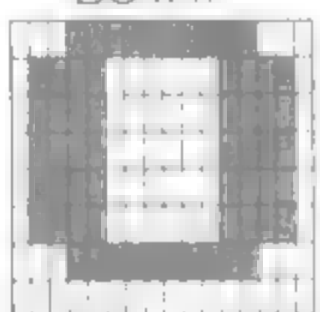
BC60H ~
BC6FH



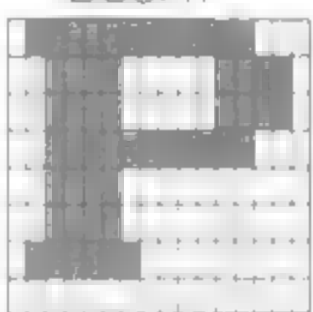
BC70H ~
BC7FH



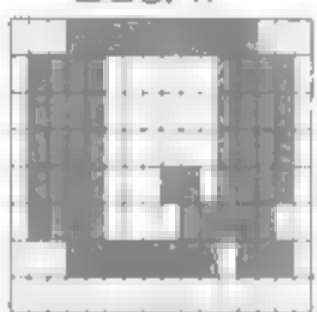
BC80H ~
BC8FH



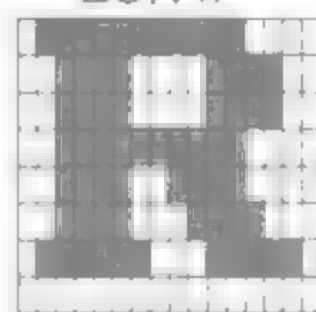
BC90H ~
BC9FH



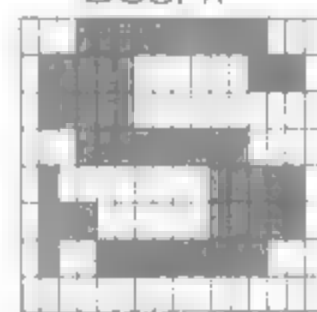
BCA0H ~
BCAFH



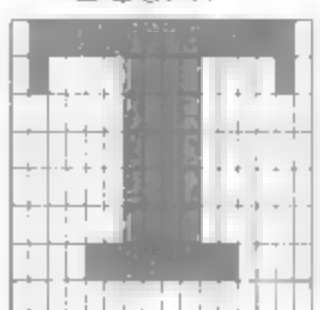
BCB0H ~
BCBFH



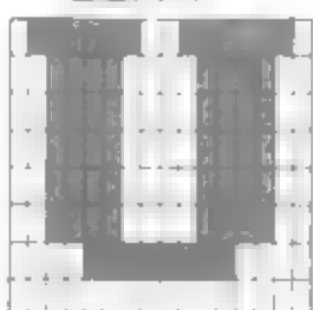
BCC0H ~
BCCFH



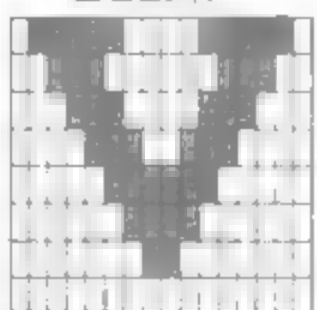
BCD0H ~
BCDFH



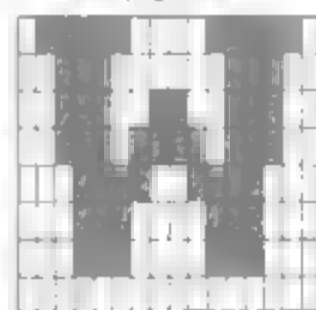
BCE0H ~
BCEFh



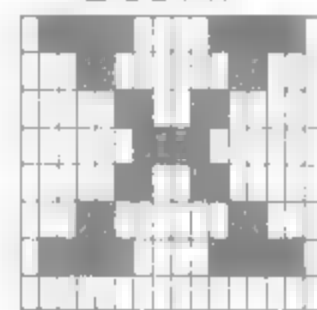
BCF0H ~
BCFFH



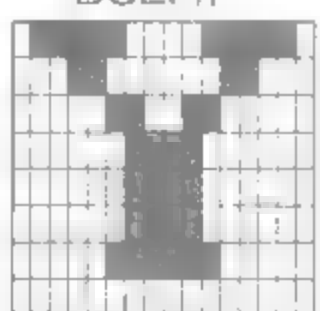
BD00H ~
BD0FH



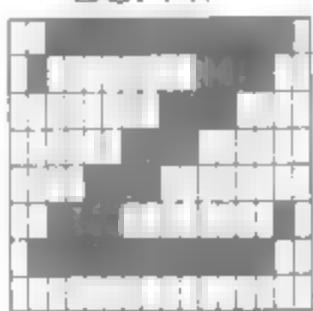
BD10H ~
BD1FH



BD20H ~
BD2FH



BD30H ~
BD3FH



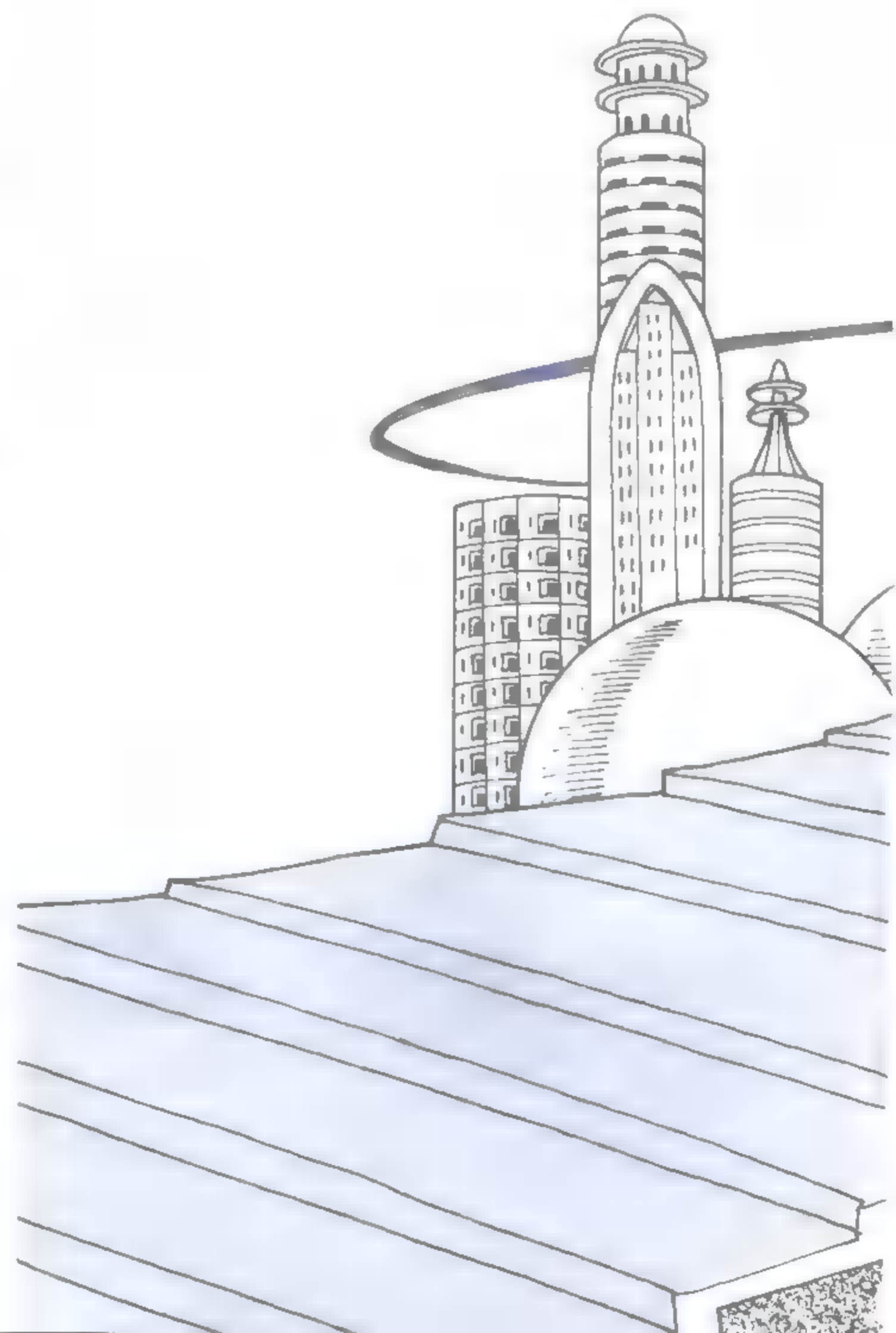
BD40H ~
BD4FH

●ウォーミング・アップ

1. 小道具…これだけはそろえておこう!
2. 数…二進数と十六進数について
3. アセンブラ…マシン語開発ツール
4. メモリーマップ…ハードウェアについて
5. 命令…ニーモニックとレジスタ
6. プログラム…その作成と実行

●BASIC に限界を感じ、マシン語を覚えようとしている今のその気持、最後まで大切にしてください。その気持さえ忘れなければ、もうマシン語なんてモノにしたも同然ですから、あせらずに気楽に進みましょう。何事もゆとりが肝心です。やさしいことも、あわてるとむずかしく見えるものです。マシン語も同じです。あまり、むずかしく考えると途中で挫折してしまいます。でも、もし■長く挫折してしまったら、その時はお手紙ください。復活の魔法をかけてあげましょう。

●P.S. マシン語なんてやさしい!! そう思ってください。ただし、すべてのマシン語プログラムがやさしいとは言いません。それは、BASIC でも同じことでしょう。そこで、BASIC を覚えた時のように、簡単なことでも1つ1つ■をしながら、その内容を理解していけばいいのです。どうか、1週間や2週間で本書の内容を読破しようなどというハリキリ精神は捨ててください。…挫折の元です。あわてなくても、ゴールは1ページずつこちらに近づいてきます。





1. 小道具…これだけはそろえておこう！

「サア〜、マシン語をマスターするぞ!!」と、期待して本書を開いた方は、ガッカリするかもしれませんが、まずは肩慣らし、ウォーミングアップです。といっても、ここで体操をするわけではなく、マシン語プログラムを組むための予備知識を、まずは頭に入れてもらおうということです。お風呂に入るのに、裸になっていきなり湯船に飛び込む人はいませんね。普通は湯加減を見たり、体を洗ってから入るはずですよ。ほんの少しの時間を惜しんで、風呂で火傷を負ったりしては、一生笑い者です。マシン語を勉強したけどわからない、という人は大抵いきなりマシン語の中にドボン…というケースが多いようです。

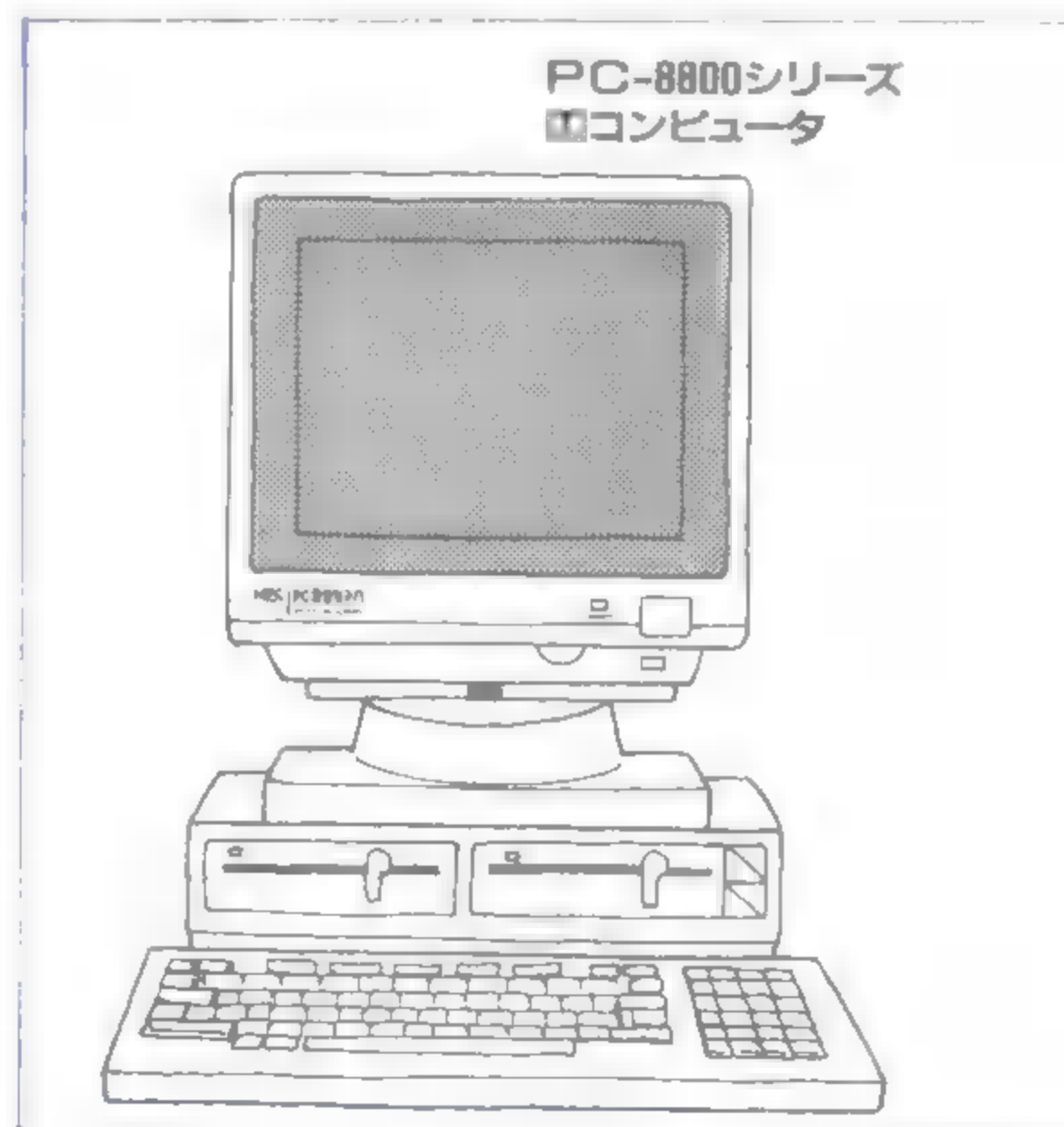
さて、マシン語を操るには、やはりそのための道具(ツール)が必要です。これは、いくら優秀な大工さんでも、ノコギリやカナヅチがなければ木を切ったり釘を打ったりできないのと同じこと。では、マシン語でゲームを作る時に利用するツールを右に紹介しておきます。

ここにあげたもののすべてが、今すぐ必要ではありませんし、本書を読むだけでマシン語を理解するのであれば、何も用意しなくても間に合うかもしれません。しかしプログラムを応用したり、自分自身でプログラムを組む場合には、それぞれが役に立つものばかりですから、財布と相談しながら手に入れるようにしてください。アセンブラに関しては、本書では、巻末のダンプ・リストを打ち込むだけで利用できる MF

-ASM 2 を利用することを前提に話を進めていきますが、他のものでも問題はありません。ただし、取り扱い方法はそれぞれ違いますので、マニュアルをよく読むことが大切です。参考書として PC-Techknow 8800, PC-8801mk II SR テクニカルメモ, N₈₈-BASIC 解析マニュアルは、お勧め品です。またここにあげた本以外にも、たくさんありますので、自分にあったものを本屋さんで捜してみてください。

なお、PC-8801mk II グラフィック・ワークブックもよろしく…(編集部)。

さらに、参考ゲーム『マジック・ガーデン』は、本書のテクニックを現実に利用した例として、またゲームとして大いに楽しんでもらえるものです。ぜひとも、あなたのライブラリーに加えておいてください…(作者談)



2. 数…二進数と十六進数について

コンピュータは人間が必要に迫られて作り出したものですから、そこには当然のことながら、人間にはできない能力を秘めています。それが、計算の速度であり、正確さであるわけです。お隣の国、中国ではコンピュータのことを電腦というそうですが、何となく人間臭さを感じて、親しみが沸いてきます。まるで、人間の頭に電気を通したみたいですが、人間の頭脳とコンピュータの頭脳との、一番大きな違いは何かといえば、コンピュータには大体とか、適当にという感覚がないことです。中流なんてないのです。つまり何でも白か黒かははっきりさせるということです。

この《アルかナイか》を数字で表現すると《1か0》になります。これが2進数の基本です。そしてコンピュータは、この《1か

0》を電気が通っているいないかで処理するのです。これは、どんなメーカーのどんな機種でも、コンピュータである以上変わらない共通点です。

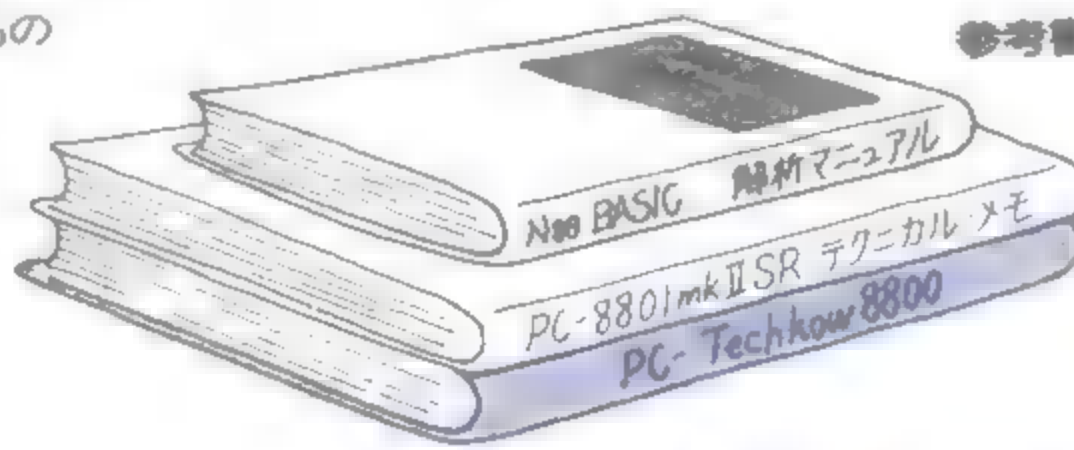
これまであなたが使ってきた BASIC にしても、最終的にはこの《1か0》の命令に内部で変換されて動いています。この内部の変換を1行ごとにするため、時間がかかり、BASICは遅いということになってしまいます。これを早くするには、最初から《1か0》で命令を出せばいいということになります。これがこれから覚えようとしているマシン語の実体です。もちろん、《1か0》だけでは2通りの命令しか作れませんから、これをモールス信号のようにいくつも組み合わせることにより、1つの命令を作るのです。そうすれば、《1か0》だけでも

2進, 10進, 16進
の計算ができるもの

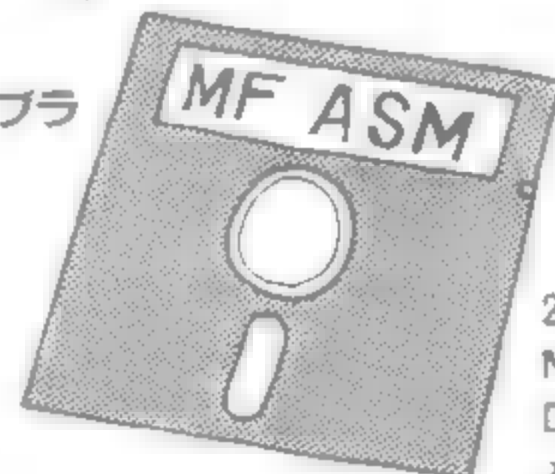


電卓

参考書



アセンブラ



2pass方式のもの
MF ASM(アスキー)
DUAD-88D(アスキー)
など

ゲーム



マジック・ガーデン(エニックス)
テグザー(アスキー)
など

筆記用具



たくさんの命令を作ることができることになります。大変なことのようですが、むずかしく考える必要はないのです。この《1か0》で作った命令を暗記しようというのではありませんから…。

ここでは、命令の基本となる《1か0》を数えるのに、ビットと呼ぶ単位を用います。ですから《1か0》が2つならば2ビット、5つならば5ビットということです。しかし命令によって2ビットを使ったり5ビットを使ったりするのは、いくらコンピュータでも処理しにくいのです。第一、どこまでが1つの命令なのかわかりません。そこで8ビットを1セットにして、これで1つの命令を表わすことにしたのです。そして、この1セットつまり8ビットのことを1バイトと呼びます。

ところで、我々が使っている NEC の PC-8801 は、ご存じのように8ビットのコンピュータです。これは、何を意味しているかということ、1度に8ビットの処理ができ

るということです。左下図を見てください。

ここにある《1か0》8つの組み合わせでできた1つの命令を、まとめて1度に処理できるのです。これは実に都合がいいですね。では、ためしに命令を1つ書いてみます。

11001001

どうですか。これはもう立派なマシン語の命令です。しかし、いくら1度に処理してくれるといっても、これではあまりに長たらし過ぎます。それにこんな命令では命令する我々の方がたまりません。そこで、まず短く表わすことから考えてみましょう。長くなった原因は、命令を2進数つまり1と0だけで書いているからです。数える基準を変えてやれば短くてすむはずですよ。

例えば、机の上に猫が17匹こっちを向きゴロニャンしているとします。この猫をもし16進数で数えたらどうでしょうか。16進数では、16まで数えて桁が繰り上がり2桁の10という数になります。10進数で17匹の猫を16進数で数えると11匹ということになります。しかし実際に机の上にいる猫の数は、変わってはいません。数える基準を変えただけで猫そのものには何もしていないのですから、これは当たり前のことです。そして、それにはこの16進数に変えるのが一番都合が良いのです。

なぜ、慣れた10進数でなくて16進数が良いのか、その前に16進数の数え方を覚えてください。当然、10になるまでは1桁で数を表示しなければなりません。

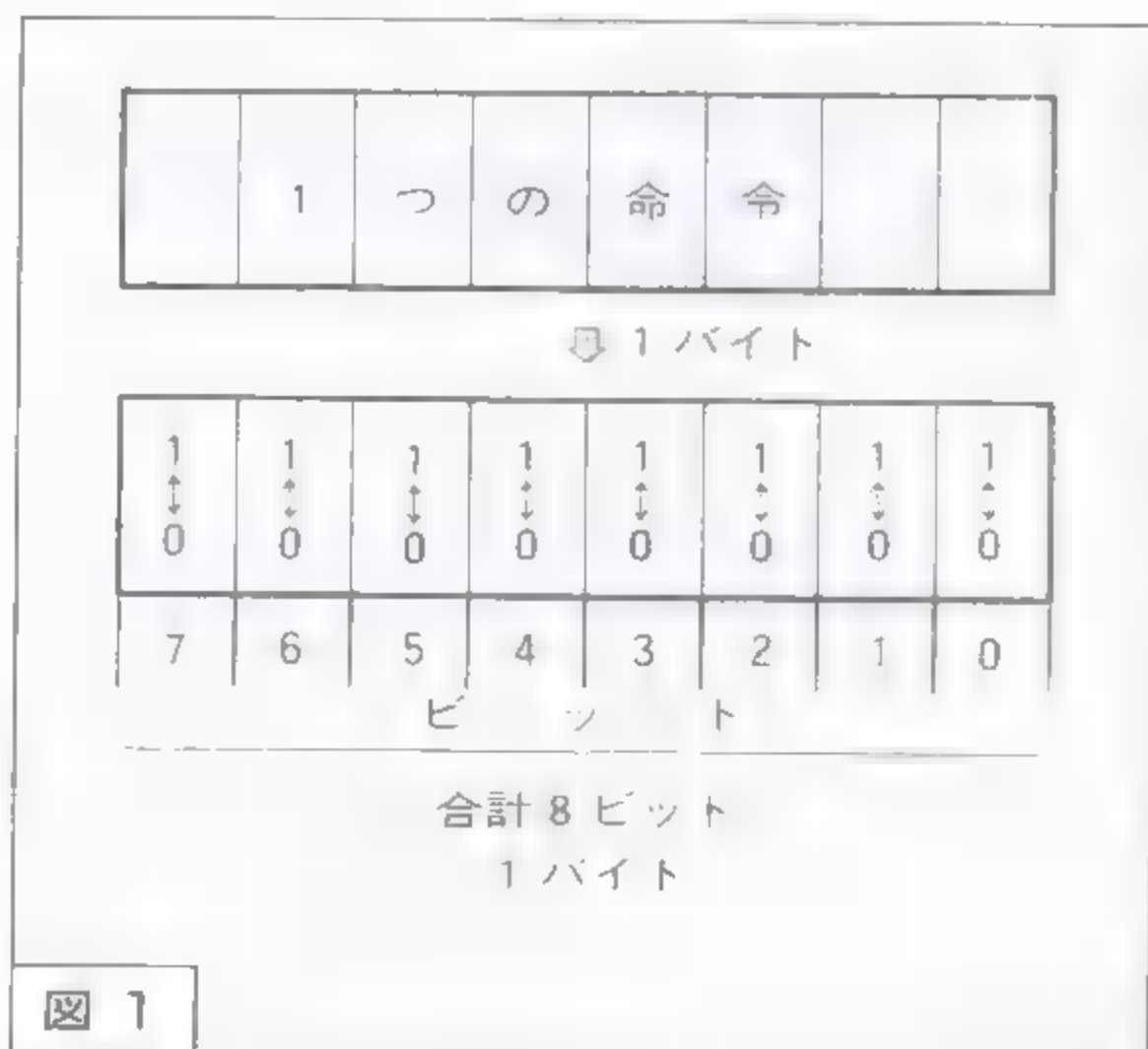
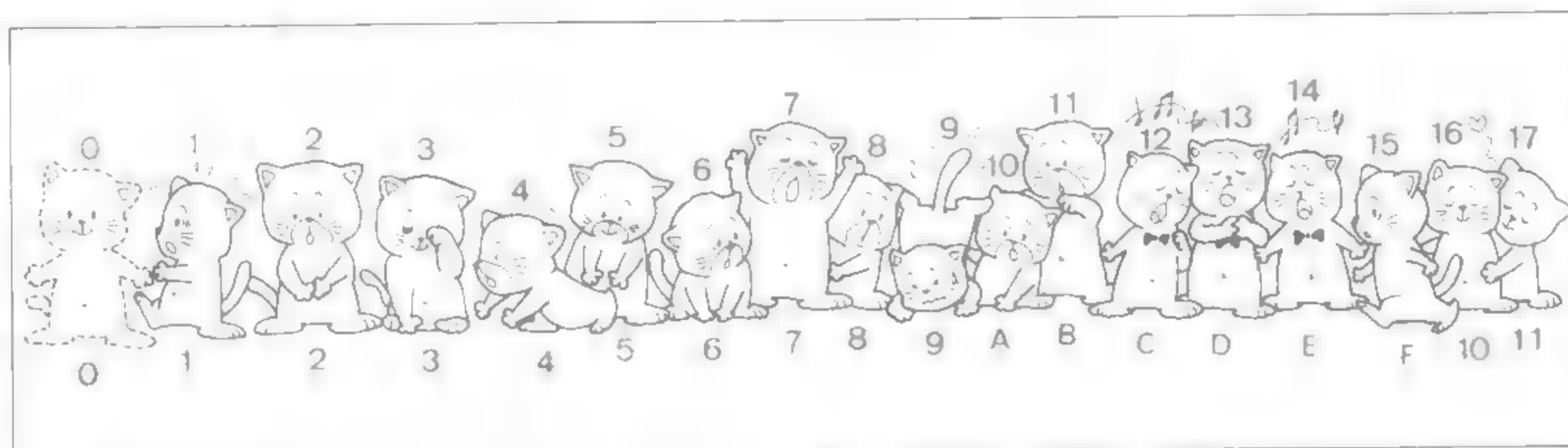


図 1



16進数はこのように表わされます。Fの次は、G…ではなく、桁が上がって10になります。そして2桁の最大数はFF。その次はまた、桁が上がって100になるわけです。

ここで8ビット(8桁の2進数)で表現できる命令の数を調べて見ましょう。1ビットにつき《1か0》の2通りの表現しかできませんから、次のように計算できます。

8ビットで表現できる命令

$$\begin{aligned}
 &= 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \quad \text{通り} \\
 &= \quad 16 \quad \times \quad 16 \quad \text{通り} \\
 &= 256 \quad \text{通り}
 \end{aligned}$$

もう、おわかりかもしれません。8ビットの16進数で表現すれば、10進数の0~255を2桁の数字(0~FF)で表わせます。

これで、先ほどの長い命令も、2桁の数字で表現できることになりました。では、16進数に早速変えてみましょう。電卓を出してください。紙と鉛筆で計算しても構いませんが、我々の目的はその計算方法をマスターすることではありません。ここは、結果だけを求めて、さらりと通り過ぎてしまいましょう。

まず、2進数のモードにして11001001と入力します。そして、16進数のモードに変

換します。C9と表示されています。

$$11001001 = C9$$

どうです。大分、スッキリしましたね。コンピュータには、この2桁の16進数で命令すれば良いのです。そして、これが我々の作ろうとしているマシン語の命令なのです。命令といっても口でするわけにはいきませんから、これをメモリに置き走らせなくてはなりません。

もう、あなたはマシン語とは2桁の16進数(00~FF)のことで、コンピュータはそれを8桁の2進数の命令とみなして実行している、ということが理解できています。それでは、マシン語を覚えるということは、この数字の意味を全部覚えるということなのではないでしょうか。もし、そうであったなら…

マシン語はやさしい』というのは、『記憶力抜群の人には…』という条件文つきの話になってしまいました。これでは、本の名も『ペテン語入門』とでもした方がよさそうです。実は、もっとわかりやすくマシン語でプログラムが作れます。そして、そのためにアセンブラというものが要なのです。

3. アセンブラ…マシン語開発ツール

マシン語の命令とは、一体どんな内容だと思いますか。かなり色々な意味を持った命令がありそうですね。ところが、実際は非常につまらないことしか命令できないのです。簡単にいえば、数字をもてあそぶだけなのです。それも2つの数を足したり引いたり、どっちが大きいかに比べたり、メモリのどこかに数字を置いてみたり…もちろん、プログラムですから比較した結果でBASICのGOTO文のようにどこかへジャンプすることもあります。それでも、行った先でまた同じように数字をいじっているだけなのです。

この程度なら、簡単に覚えられそうな気がしませんか。しかし、次の命令を見てください。左側の16進数がマシン語です。右側がその意味です。AとかBとかいうのは変数と思ってください。

47…BにAの値を代入(B=A)

4F…CにAの値を代入(C=A)

78…AにBの値を代入(A=B)

79…AにCの値を代入(A=C)

4つとも似たような内容なのに、マシン語の数字はバラバラです。その上、この数字を見ただけでは、代入するとかAとかBとか連想することは全く不可能です。となると、ただ丸暗記をするしか覚える方法はないそうです。まあ、世の中には平気でこの数字でプログラムを組む人もいますが、今はコンピュータの時代です。

そんな面倒なことは、コンピュータにまかせましょう。我々は、もう少しわかりやすい記号で命令を書いて、それをコンピュータで数字に変換してもらえばいいのです。この数字に変換してくれるソフトのことをアセンブラといいます。そして、我々にわかりやすい、この記号のことをニーモニックというのです。このニーモニックはマシン語の数字を人間にわかりやすく記号化しただけで、その意味や内容はまったくマシン語と同じですから、これもマシン語と呼ばれます。あなたは、これからこのニーモニックのマシン語をマスターし、プログラムを組んでいくのです。

これで、なぜ貴重なお金を出してまでアセンブラが必要か、何となくわかったのではないかと思います。といいつつ、実は、PC-8801には、最初からアセンブラ機能がついているのです。と云ったら怒るでしょうね。ただ、このオマケのアセンブラでは、長いマシン語プログラムを作ることか、まず不可能なので誰も作ろうとしないからです。

このアセンブラは、ワン・ライン・アセンブラなのでスクリーンエディットができないのです。苦心して作った長いプログラムに、何かバグが見つかったとします。その度に、間違いがあった所から、もう一度、全部のプログラムを書き直さなければならないとしたら…。さらにラベルが使えないとか、ニーモニックがZ80用でなくインテル8080用なので命令の違いや使用できな

『DUAD-88D』の特徴

1. 本格的なスクリーン・エディット機能を備えている。
2. アセンブラには、ラベルのソート出力、クロスリファレンス・リストなど豊富なオプションがついている。
3. プログラム移植、解析に便利な多機能型逆アセンブラがついている。
4. ディスク上にアセンブリされたプログラムをオフ・セットをつけてロードできる。
5. リロケータブルなデバッキング・ツールがついている。
6. 本格的なため少々値段が高い(¥49,800)。

『MF-ASM2』の特徴

1. とにかく値段が安い(本書付録のダンプ・リストを打ちこめばタダである)。さらに本書のプログラム・リストがセーブされているディスクアルバム 10 に入っているし、テープ版の『MF-ASM』も市販されている。
2. ソース・プログラムでは BASIC の REM 文として作成するので、BASIC の感覚でスクリーン・エディットができる。
3. 操作を BASIC 上で行なうので、初心者でも扱いやすい。
4. 本格的な大プログラムを組むには力不足であるが、ゲーム作成用アセンブラとして使うには十分である。

い命令があるとか、開発用のアセンブラとしては、不適當といえます

しかし、短いテスト・プログラムの作成や簡単な変更、あるいは、プログラムの見直しなどには、大変便利なものですので、その目的で利用すればそれなりに価値のあるものです。使用方法については、PC-8801 本体附属のマニュアル(モニタの所)に詳しく書いてありますので、そちらの方をお読みください。

『DUAD-88D』と『MF-ASM』について、簡単にその特徴と違いを上表としてまとめておきます。

本書では、できるだけ多くの方にマシン語をマスターしてもらうためには、『MF-ASM』のグラフィック対応版『MF-ASM2』を

巻末に載せ、これを基準にして説明をしていきます。すでに、『DUAD-88D』やその他のアセンブラをお持ちの方は、それを利用できるのは、いうまでもありません。



4. メモリーマップ…ハードウェアについて

マシン語の命令をコンピュータに実行させるということは、メモリに16進数の命令(00-FF)を置いて、そこを走らせることだということは既にも書きましたが、メモリとは文字通りその数を記憶する場所のことです。記憶するだけですから、00-FFの数であれば、別に命令でなく何かのデータでも構わないわけです。第一、メモリ自身は命令なのかデータなのか判断できないのです。ただ、1バイト(00-FF)の数を記憶しているだけなのです。このあたりの実直さは、いかにもコンピュータらしいといえるかもしれません。

それでは、走るのはいく体誰なのでしょう。そして、命令とデータの区別はどのようにしてつけているのでしょうか。これは、ハッキリさせておかなければならない問題です。走るのは、もちろん人間ではありませんね。CPU (CENTRAL PROCESSING UNIT の略) というコンピュータの心臓部に当たるものです。このCPUがメモリの数を読み取って、その命令を実行するわけです。しかし、CPUもメモリの数が命令なのかデータなのかの判断はできません。では、一体誰がどこでその判断をしているのでしょうか。それをできるのは…この世でただ1人、プログラムを作ったあなたしかいません。つまり、CPUが命令の所だけを走るように、プログラムを組んでやらなければいけないということです。もし、データの所を走らせたら…その時は、まず、まちがいなく暴走します。たいていは画面

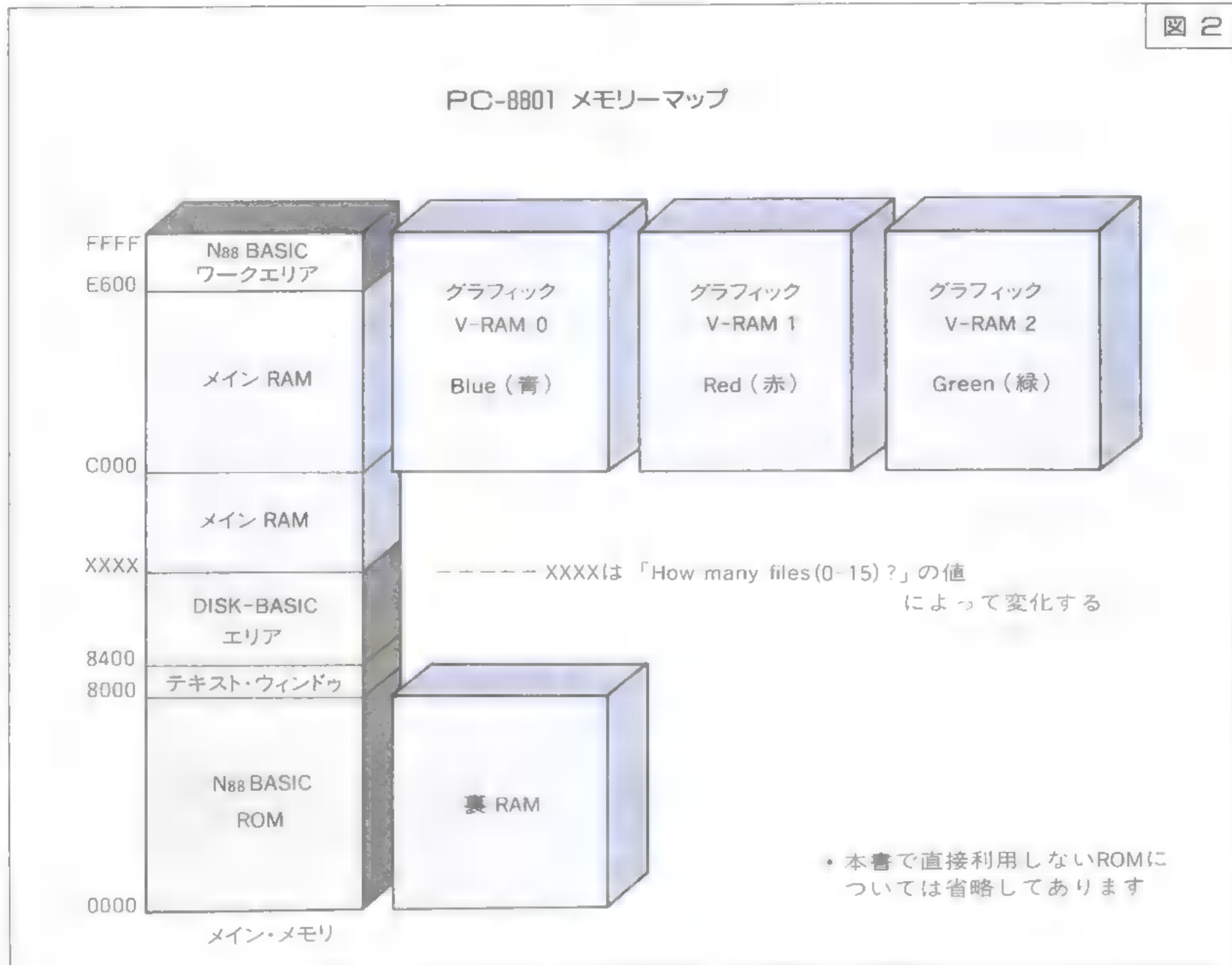
がメチャクチャになって、2度とキー入力できなくなります。こうなったら素直にリセットする以外に道はありません。

そこでCPUが暴走しないようなプログラムを組むには、メモリを我々がきちんと管理する必要がでてきました。それには、PC-8801のメモリがどういう構成になっているのか知らなければなりません。メモリというのは単なる記憶場所ですから、数は理論上いくらでも増やすことができます。しかし、その上を走るCPUに限界があるのです。8ビットのCPUの場合、最大でも64Kバイト(1Kバイト=1024バイト)のメモリ空間の中しか走り回れない(アクセスできない)のです。

CPUが直接アクセスできる最大メモリ数
 =64K バイト
 =65536 バイト
 =10000 バイト(16進数)

これだけのメモリを管理するには、まずメモリの区別がつくようにしておかなければなりません。それには、1つ1つのメモリに番号をつけて、番号の小さい方から順に並べていけばいいのです。メモリの総数は、16進数で10000バイトあるわけですから、0000からFFFFまでの番号をつければ、それぞれのメモリの区別ができるということになります。そして、このようにしてつけられたメモリ番号のことを、アドレス(番地)といいます。CPUはあなたに命ぜられたスタート・アドレスから、その中にある命

図 2



令を1つ1つ読み取り実行していくわけです。

今までの説明に比べ、上図のメモリーマップは何だかゴチャゴチャしていて変な感じがします。この理由はPC-8801のメモリが64Kバイトではなく、184Kバイトもあるためです。そのため、PC-8801ではバンク切り換えという方法を用いて、全部のメモリ空間をCPUがアクセスできるようにしているのです。例えば、アドレスでC000-FFFF番地というメモリ空間は、メイン・メモリの他にも3種類があります

(図2参照)。どのメモリ空間でもスイッチを切り換えることにより、メイン・メモリ空間と入れ換えることができるのです。これをバンク切り換えと呼びます。

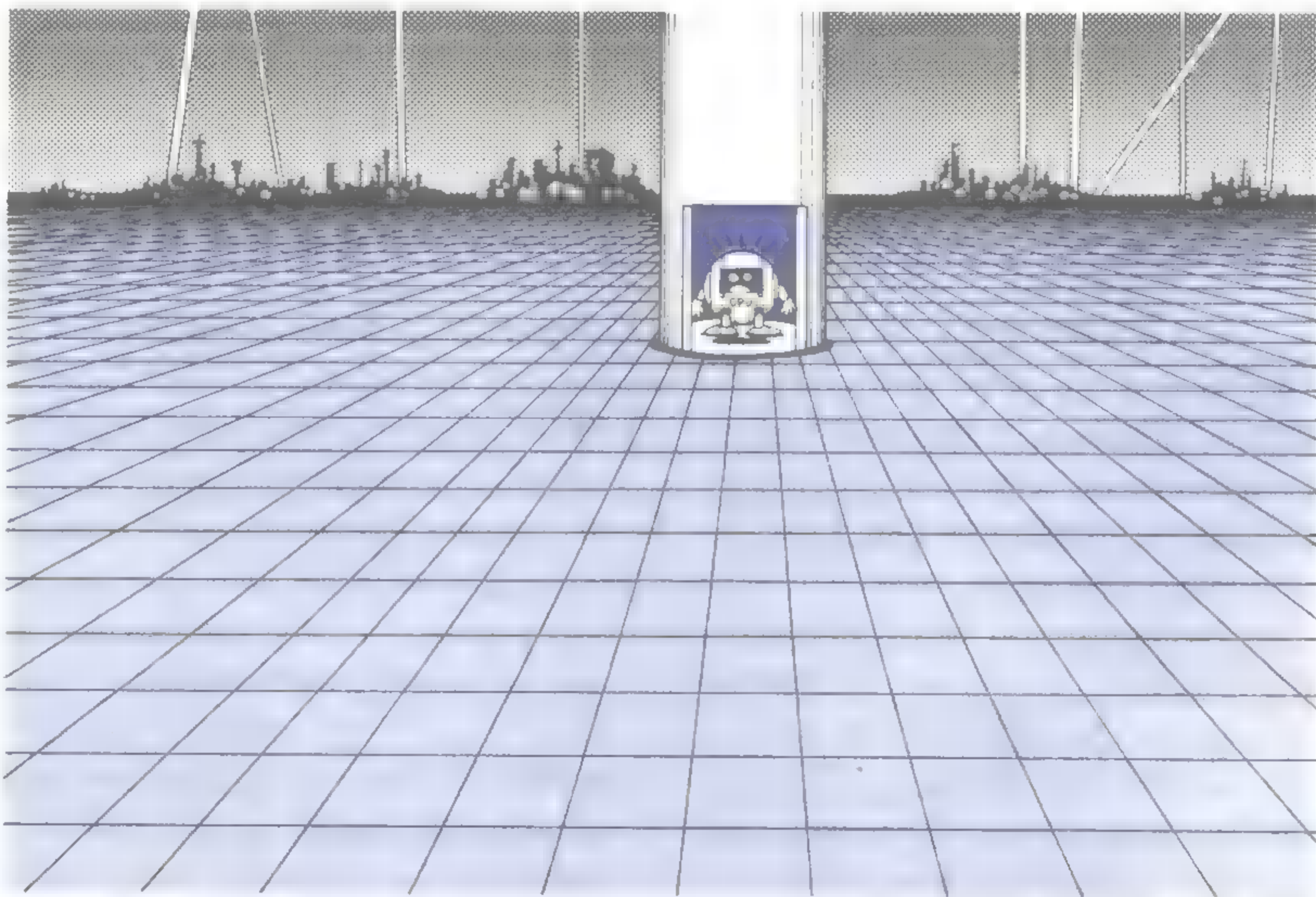
これだけのメモリ空間の中で、我々がマシン語のプログラムやデータを置くことができるのは、ここでは《CLEAR文が成立する次の番地からE5FF番地の間》と考えてください。つまり、BASICで『CLEAR, & HB4FF』と宣言すれば、《B500-E5FF番地》まではあなたが自由にプログラムやデータを置けるエリアであるということです。

CLEAR 文については、PC-8801 のマニュアルに詳しい説明がされています。また、CLEAR 文の宣言をしないと、E500 番地台の後半はプログラムが置けませんから、マシン語プログラムには必ず《CLEAR 文の宣言をする》というように覚えておいた方がいいでしょう。

それから、0000-7FFF 番地にある裏 RAM は、バンク切り換えによる使用もできますが、8000-83FF 番地のテキスト・ウィンドウを通して読み書きができます。このエリアは、裏 RAM の中の任意の 1K バイトに、命令を書いたり読んで実行できる特殊な窓なのです。なぜ、こんな窓があるかというと裏 RAM とは本来、BASIC プログラ

ムを置くためのメモリなのです。しかし、バンク切り換えをすると、N_{RR}-BASIC ROM 自体が切り離されて使えなくなってしまうため、このような窓を通してバンク切り換えなしでも裏 RAM の BASIC プログラムを読めるようにしてあるのです

メモリーマップについては、最初に紹介した『PC-Techknow 8800』などの参考書にワーク・エリアのことなども含めて詳しく説明されていますので、ここでは全体のメモリの構成を把握するだけにします。あまりマシン語の準備ばかりしていて、本筋に中々はいらないと、折角のあなたのヤル気がなくなってしまうかもしれませんからね…



5. 命令…ニーモニックとレジスタ

イヨイヨ、マシン語本体にたどり着きました。プログラムを組む前にまず命令にはどんなものがあるのか、軽く見ることにしましょう。Appendix 2 のマシン語のインストラクション表を見てください。ここにあるニーモニックと書かれた記号が、これから使うマシン語です。むずかしそうに感じるかもしれませんが、とりあえずニーモニックとはどんなものか、何種類位あるのか、それだけでも確認してください。今までニーモニックとは、我々にわかりやすい記号とだけしか書かれていませんでしたが、この表からその記号が○とか△ではなくアルファベットだということがわかりました。そして、実はこのアルファベットは英語の単語を省略したものなのです。このことがニーモニックが人間にわかりやすいという理由なのです。

ここで、次の文字を覚えてください。

A B C D E H L

これはマシン語で使える変数です。マシン語の場合、BASIC のように自由に変数をつくることはできません。しかし、7 つでもうまくヤリクリすれば何とかなるものなのです。これらの変数には、それぞれ1バイトの数値(00-FF)を記憶することができます。また、BC, DE, HL, はペアで使うことにより16ビットのデータを処理できるのです。そして、これらの変数のことはレジスタと呼ばれます。この7つのレジスタは一見、同格のように見えますが、それぞれ

能力に差があり、中でもAレジスタは計算の命令が他のレジスタより多いので、アキュムレータと呼ばれます。またBC, DE, HLなどのペアになったレジスタはペアレジスタと呼ばれます。本当はこの他にもレジスタと呼ばれるものはあるのですが、今はこれだけ覚えてください。

マシン語の命令というものは、そのほとんどがレジスタに関係があります。ということは、まずレジスタに数値を代入する命令を知らなければなりません。

LD A, 0D5H

これはAレジスタに16進数のD5をロード(代入)するという意味です。LDはLOADの略です。LOADと言ってもBASICのLOAD命令とは違うので注意してください。D6の前後に変なものがついていますね。これは16進数を表示する時の決まりで数字の最後にはHを、また数値がA-Fで始まる場合には頭に0をつけなければならないというきまりです。本書でも、ここから先は16進数の最後にH(16進数:Hexadecimal)をつけることにしますが、頭の0は本文中では邪魔なのでプログラムにだけつけるようにしました。同じ書き方で、他のレジスタにも数値をロードできます。

例 LD B, 17H ; Bに17Hをロード
LD E, 0F3H ; EにF3Hをロード
LD HL, 0A123H
; HLにA123Hをロード

ニーモニック中のスペースは1スペースあればいいのですが、そろえると後で見やすいので、**TAB**を用いて整然と書く習慣をつけてください。

また、数の表記については、16進数以外にも10進数やマイナスの数、そして加減算を含んだ式の状態で書くこともできます。

これらは、アセンブルする時に、自動的に16進数に変換されることになりますが、具体的な例については本書での使用例を見て確認することにしましょう。

このLD命令というのは一番多く使われる命令です。下にその例を示します。

1. あるレジスタの値を別のレジスタへ移す。移す側の値は変わらない。

```
LD  A, B      ; AにBの値をロード
LD  D, L      ; DにLの値をロード
```

2. 指示された番地に入っている値をロードする。番地の中身は変化しない。

```
LD  A, (0B300H) ; B300H番地にある値をAにロード
LD  A, (BC)     ; BCレジスタで示される番地にある値をAにロード
LD  HL, (0D500H) ; D500H番地にある数値をLに、D501H番地にある数値をHにロード
```

このようにカッコで囲むと、その番地の中にある値を意味します。また、3番目のようにレジスタペアにアドレスの中から数値を入れる場合、入る順が逆になります。しかし、次の3.の例に示したようにアドレスの中に入れる時にも逆になりますから、実際はまったく気にする必要はありません。

3. レジスタの値を指示された番地の中に移す。レジスタの値は変わらない。

```
LD  (0B300H), A ; B300H番地にAの値を入れる
LD  (BC), A     ; BCレジスタで示される番地にAの値を入れる
LD  (0D500H), HL ; D500H番地にLの値を、D501H番地にHの値を入れる
```

以上がLD命令の主な使用方法です。要するに、ロード命令とは数値を移動するための命令であると思えばいいのです。覚えなければならない命令を書いていくと、それこそキリがありませんから、命令につい

ての説明はこれが最初で最後です。この先、プログラムでわからない命令がある場合は、Appendix 4のマシン語命令小辞典を見て理解するようにしてください。

6. プログラム…その作成と実行

これから、実際にマシン語のプログラミングをしていきますが、重要なことは必ずテストの実行をするということです。そして、なるべく自分の手でプログラムを入力してください。これが、マシン語の書き方や命令、それにアセンブラの用法を覚える一番いい方法だからです。また、『MF-ASM 2』以外のアセンブラを使用される方は、そのマニュアルをよく読んで使い慣れることが大切です。アセンブラさえ使いこなせれば、マシン語はマスターしたも同然です。

さて、『MF-ASM 2』の使用方法ですが、『MF-ASM 2』の入力方法や文法上の詳しい使用法は、Appendix 1「MF-ASM 2」の所を読んでください。ここでは、練習プログラムによる基本的な使い方だけを書いてあります。まずは、『MF-ASM 2』のプログラムをメモリ上にロードしなければなりません。これもマシン語ですから、最初に CLEAR 文でマシン語エリアの確保をします。本書においては、最後のスクロール・ゲーム用の『mapped』以外のプログラムでは、ファイルを必要としないので、『How many files(0-15)?』には必ず 0 を入力するようにしてください。これは、BASIC、マシン語共にフリー・エリアを大きく取れる必要があるからです。

```
CLEAR , &HB4FF
BLOAD " miasm 2 ", R…このプログラム
                        は Appendix 1
```

これでプログラム作成の準備ができたこととなりますが、更に自動的に注釈文にしたい場合は、

```
BLOAD "autoq", R…このプログラムは
                        Appendix 1
```

とすることにより、BASIC で AUTO 命令を実行すれば、行番号とともに『』がついてくるようになります。

テープにセーブしている方は、次のようにモニタからリード命令でプログラムをロードしUSR命令を用いて走らせてください。

```
CLEAR, &HB4FF
MON [ ]
h]R [ ]
h] ^ B… CTRL [ + B]
      miasm 2 の場合
DEF USR=&HB900 : A=USR(0)
      autoq の場合
DEF USR=&HF2E0:A=USR(0)
```

本節末の List 1-1 を作成してください。BASIC のプログラムと同じ要領で行番号を入力してからプログラム・リストどおりに打ち込めば良いのです。これが記念すべきマシン語プログラムの第 1 号です。

なお思い出にセーブしたい方は、普通に BASIC プログラムをセーブする要領でセーブすれば、いつでもソース・リストとして見たり修正したりできるわけです。

次に、

CMD 

と、入力すればアセンブルされて、グラフィック・V-RAM のグリーン面にラベル・テーブル*1が、ブルー面にオブジェクト・プログラム*2が生成されます。画面に変な模様が描かれるのは、その作業を実行している証拠です。もし、プログラムに文法上のミスがあった場合には、この段階でエラー行が表示されますので、BASIC に戻りプログラムを修正します。そして、エラーもなく無事にアセンブルが完了したら、メモリにオブジェクト・プログラムをロードします。

Option ? O 

LOAD OFFSET ? 

RETURN TO BASIC OR MONITOR (B/M) ? B

これで、マシン語プログラムが BC00H

番地にロードされたわけです。その他のオプション・コマンドについては、Appendix 1 をよく読んでください。ここで、画面をクリアしてからプログラムの実行に移ります。

CLS 2 

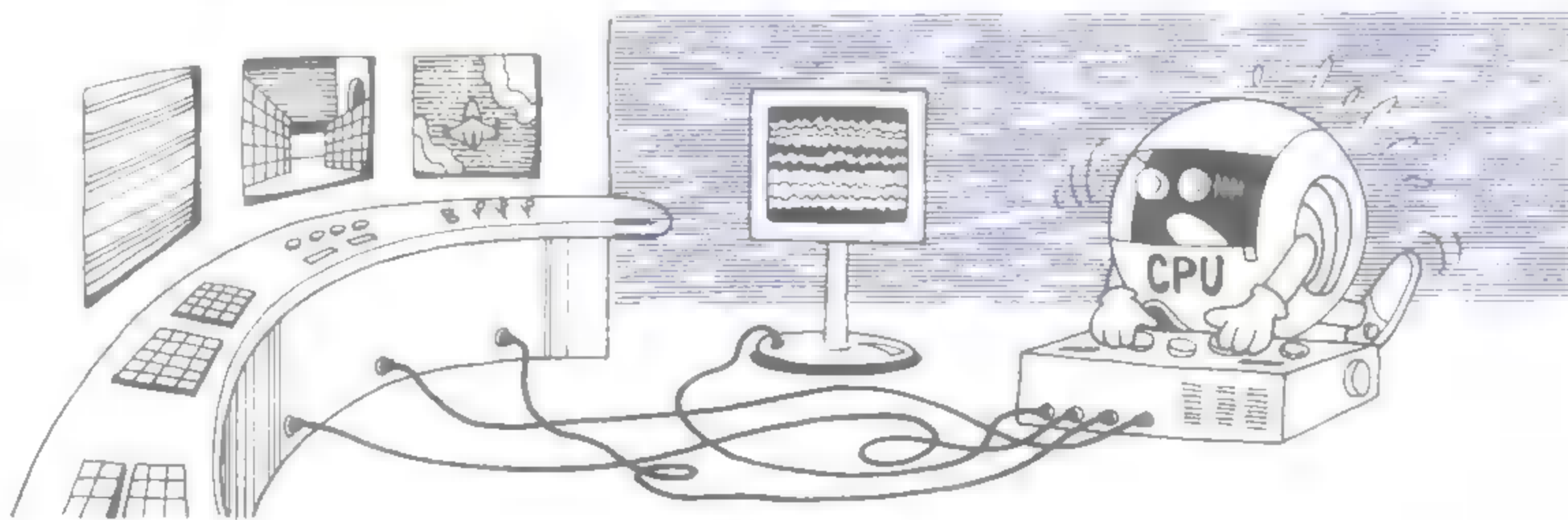
MON 

h]GBC00 

h]

画面の左上に、ほんの数ミリの白い線が描かれているはずです。これが、すべてのグラフィックの基礎で、ドットに直すと 8 ドットに相当します。これは、図 3 のようにグラフィック V-RAM は、画面を 8 ビットつまり 1 バイト単位で管理しているからです。アドレス 1 つで 8 ドット分の表示を受け持っているというわけです。

それではプログラムが、どのように実行されているのか、1 つ 1 つ確認してみましょう。まず図 4 を見て下さい。



* 1 ラベル・テーブル・MF-ASM は、2 ハス方式なので、1 ハス目に各ラベルが指すアドレスを記憶しています。この時の記憶エリアをラベル・テーブルと呼びます。

* 2 オブジェクト・プログラム：アセンブルによって、できるマシン語のことを言います。MF-ASM 2 では、アセンブルする際、このオブジェクト・プログラムをブルー面に一時置いています。

図 3

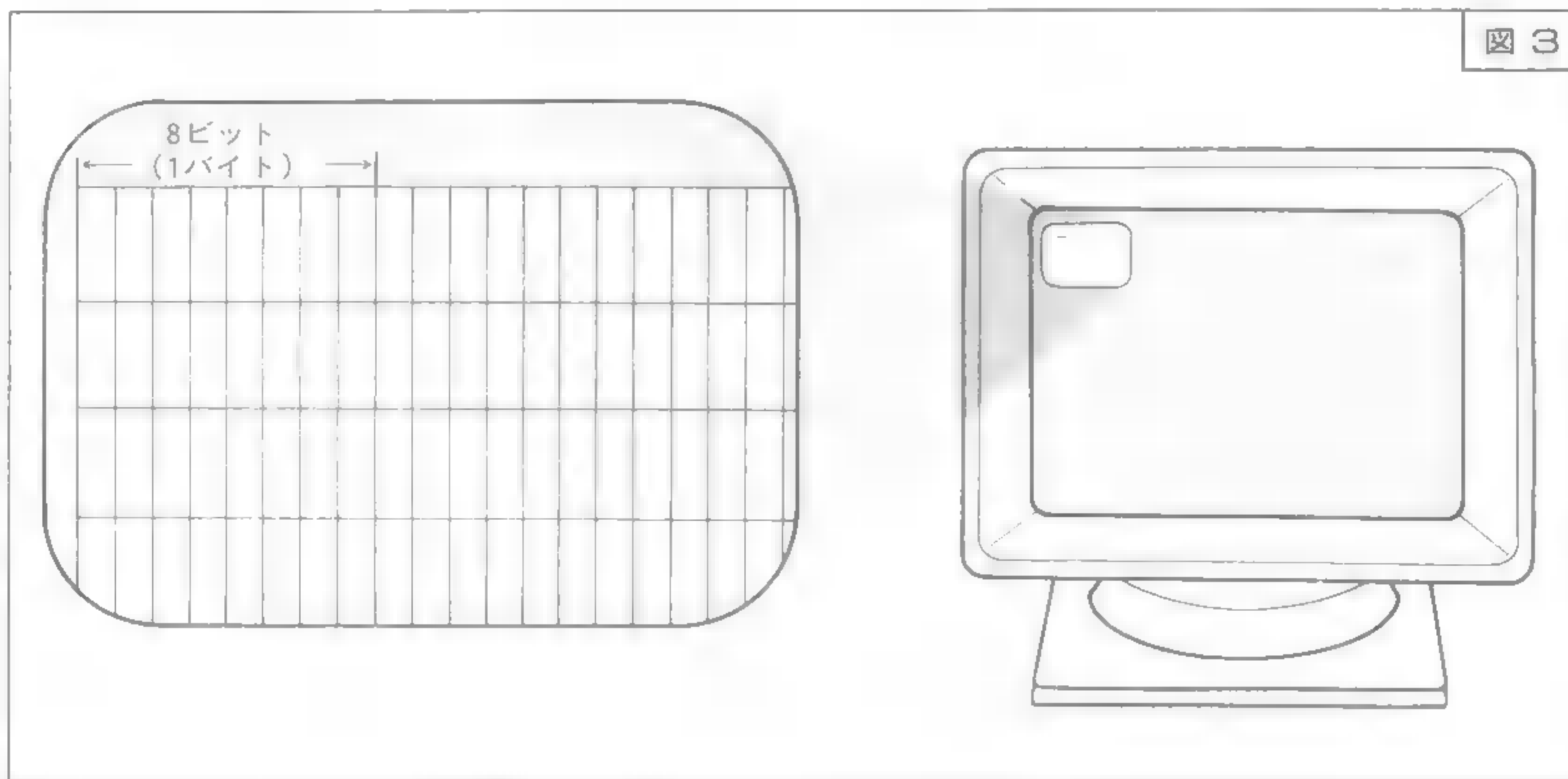


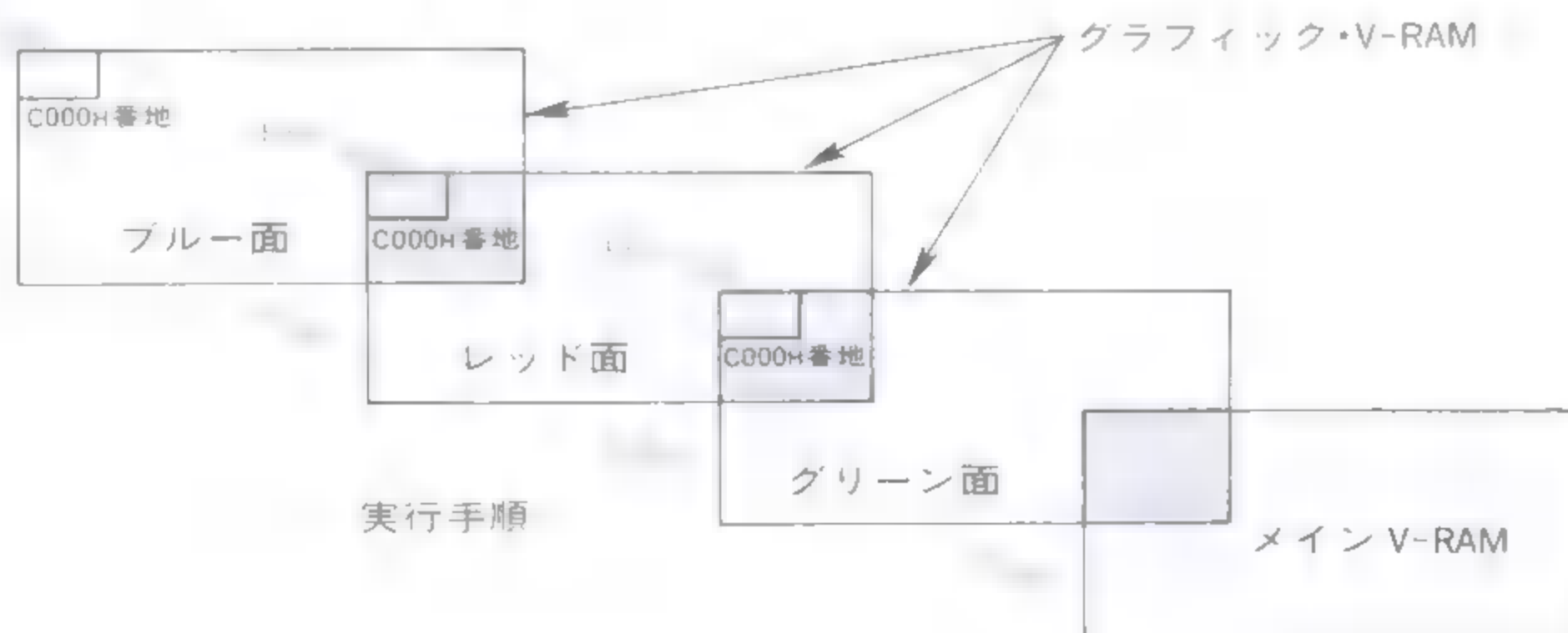
図 4

List 1-1 の実行過程

1 割込みの禁止

BASICのインタープリタでは、キースキャンなどの処理に割込みという手法を用いている。なお、プログラムは、N88BASICワークエリア(図2参照)という部分にあるため、バンク切り換えを行なうと、割込み処理プログラムも切り離され、割込みがかかると暴走してしまう。そこで、バンク切り換えをする前には必ず割込みを禁止しなければならない。

2 バンク切換え & HC000HにFFHを書く



3 割込みの許可 BASICのインタープリタを正常に動かすため

4 モニタ(h)の状態に)へ戻る

グラフィック・V-RAMとグラフィック座標の関係

座標	0	128	256	384	512	639
0	C000	C010	C020	C030	C040	C04F
1	C050	C060	C070	C080	C090	C09F
2	C0A0	C0B0	C0C0	C0D0	C0E0	C0EF
...
197	FD90	FDA0	FDB0	FDC0	FDD0	FDDF
198	FDE0	FDF0	FE00	FE10	FE20	FE2F
199	FE30	FE40	FE50	FE60	FE70	FE7F

34

このグラフィック・V-RAMのメモリ・マップと、BASICのグラフィック座標との関係は上の表のようになります。

FE80H番地からFFFFH番地までは、何も使われていないフリーエリアです。なお、表の縦の200行を省略なしで見られるプログラムを載せておきますので、利用してください。PRINT命令をLPRINTとして、1枚プリントアウトしておくと、これから先何かと役に立つかもしれません。

これで、マシン語プログラムのためのウォーミング・アップはOKです。2章では、キャラクター・パターンの表示から移動と、プログラムも急激に進みます。グラフィックの基礎固めのためにも、このアドレス表を見ながら色々な位置に白だけでなく赤や黄色などの線を引く練習をしてみてください。そして、それを2章へ進む条件ということにいたしましょう。

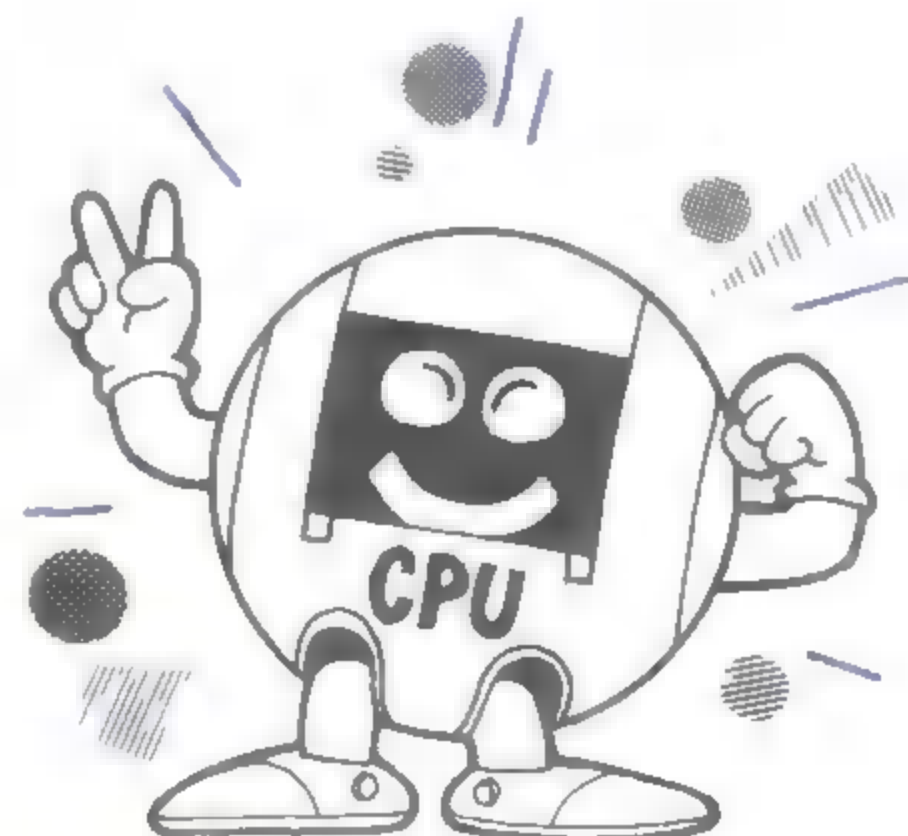
```

10000 '***** Graphic V-RAM Address *****
10010 PRINT 'DOT: 0.....128.....256.....384.....512.....639'
10020 FOR N=0 TO 199
10030   PRINT USING '###: ';N;
10040   FOR M=0 TO 79 STEP 16
10050     PRINT RIGHT$("00"+HEX$(-16384+N*80+M),4); " --- ";
10060   NEXT
10070   PRINT RIGHT$("00"+HEX$(-16384+N*80+M-1),4)
10080 NEXT
10090 END

```

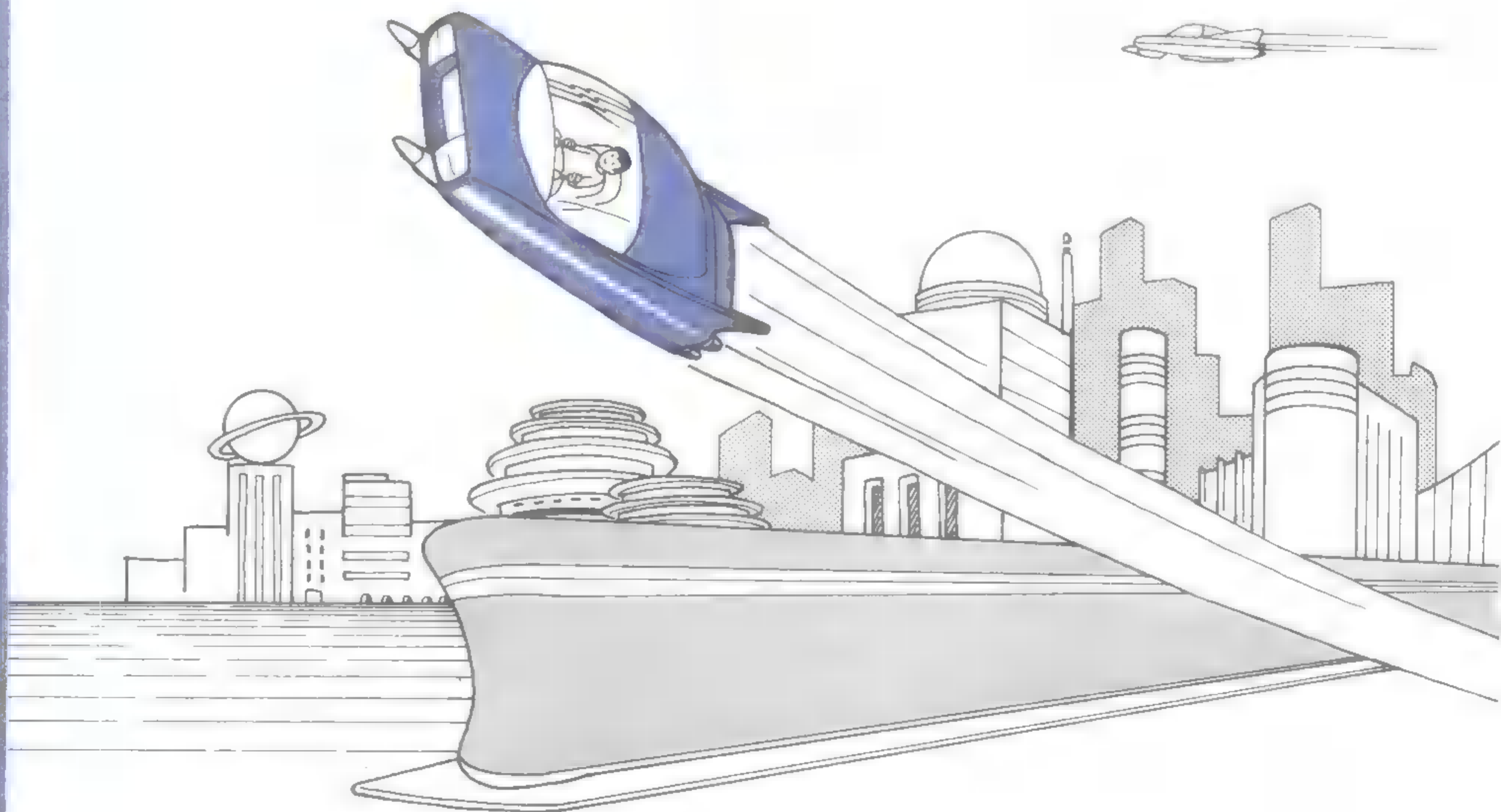

List 1-1 線を引く

10000	;	***** List 1-1 *****	
10010	;		
10020	ORG	0BC00H	プログラム開始アドレス=BC00H
10030	;		
10040	TEST:		ラベル名
10050	DI		割り込み禁止
10060	LD	A,0FFH	A ← FFH(AにFFHを代入)
10070	OUT	(5CH),A	フル画面にハंक切り替え
10080	LD	(0C000H),A	C000H番地にAの値(FFH)を入れる
10090	OUT	(5DH),A	レット面にハंक切り換え
10100	LD	(0C000H),A	C000H番地にAの値(FFH)を入れる
10110	OUT	(5EH),A	クリーン面にハंक切り換え
10120	LD	(0C000H),A	C000H番地にAの値(FFH)を入れる
10130	OUT	(5FH),A	メインRAMにハंक切り換え
10140	EI		割り込み許可
10150	RST	38H	モニタ(h)の状態へ戻る



● キャラクタ・パターンの表示と移動

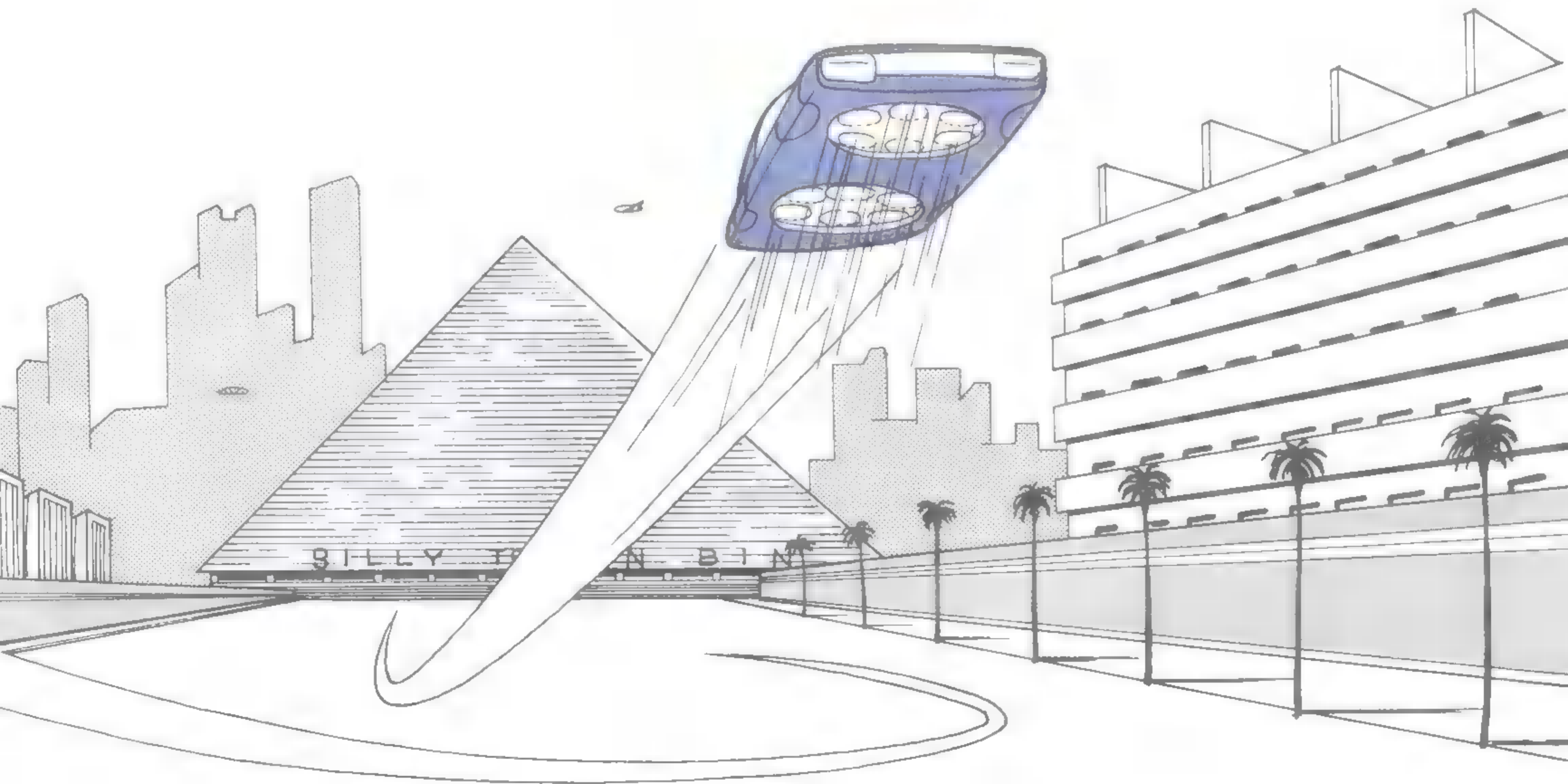
1. 座標…ゲームのためのゲーム座標
2. 豆腐…とりあえず白い四角形を表示
3. パターン…キャラクタの作成
4. パターン表示…キャラクタ登場
5. パターン消去…キャラクタを動かす前に
6. パターン移動…データにそって移動
7. 大量出現…1人じゃつまんない!
8. キー入力…コントロール & ショット



■マシン語ゲームのすばらしさは、何と言っても画面の中を高速に動き回るキャラクタです。こればかりは、BASIC ではそう簡単に実現できません。マシン語をマスターしたい一番の理由も、たいていの場合こんなところにあるのではないのでしょうか。

●「マシン語を使えば、キャラクタを思い通り動かすことができる。きっと、マシン語にはBASIC にはないキャラクタの表示命令とか、それを動かす命令があるのではないか…」

●そんな期待を持ってマシン語の命令をながめたことはありませんでしたか。そして、わけの分からない記号ばかりで、ガッカリしたのではないですか。私とマシン語との出会いは、そんな期待ハズレから始まりました。しかし、心配することはありません。この2■が終わる頃には、あなたは自分でオリジナルなキャラクタ・パターンを作り画面の中を自由に動かせるようになります。さらに、次の3章で完成する簡単なシューティング・ゲームの第一ステップでもあるのです。これは、マシン語がむずかしいと言っても、この程度のむずかしさだという証明なのです。



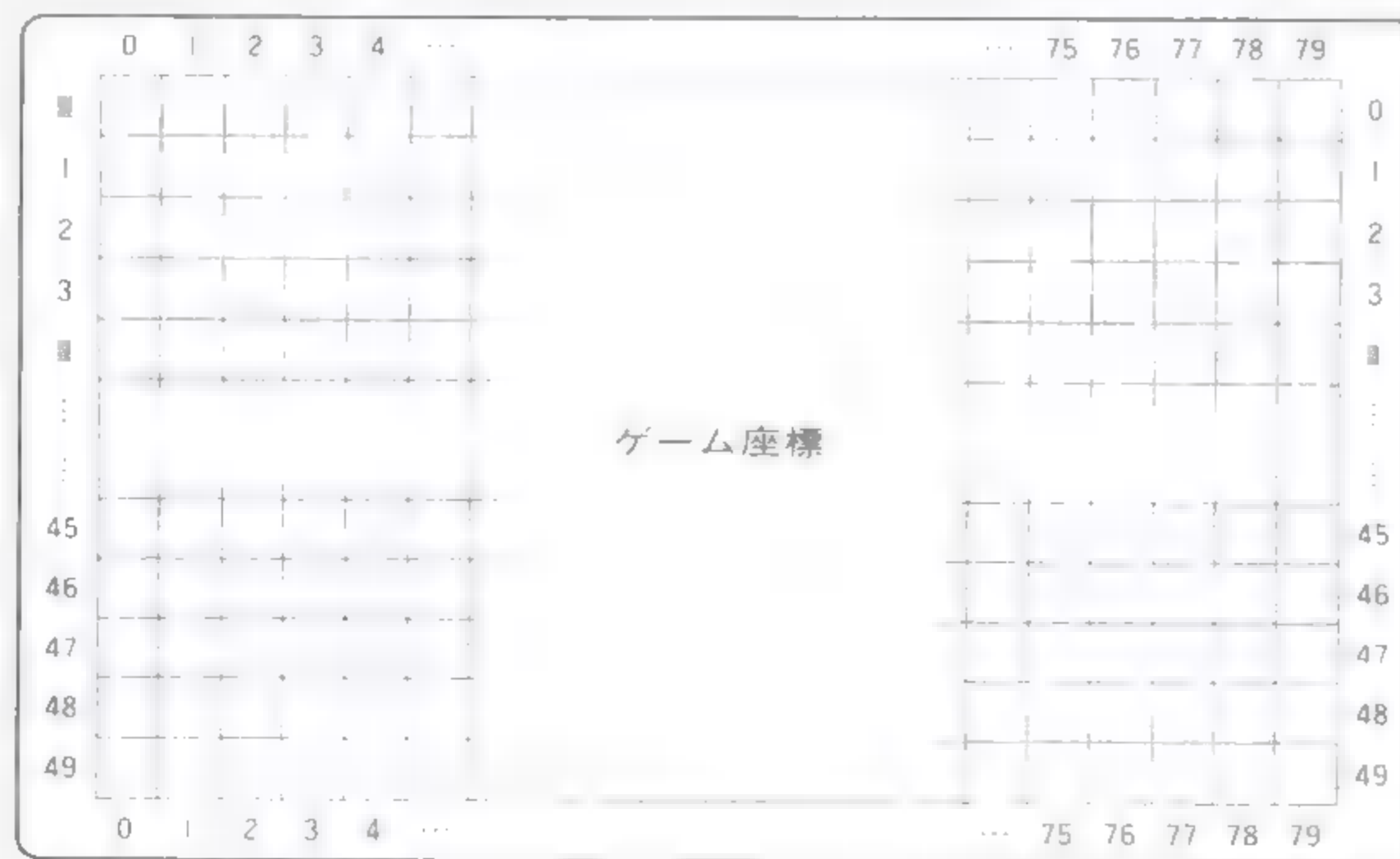
1. 座標…ゲームのためのゲーム座標

アルファベットはわずか 26 文字しかありませんが、だから英語がやさしいと考える人は、まずいません。それは、言語というものが文字を組み合わせて作られている、ということを知っているからにほかなりません。どんなにむずかしい単語でも、1文字1文字は、a, b, c, …のどれかですから誰でもわかりますが、それがまとまって1つの単語となると、その意味を知らなければ解読困難です。マシン語とは、コンピュータのアルファベットです。単独の意味がわかって、プログラム全体の意味を理解できるとは限りません。逆にいうと、マシン語でプログラムを組むということは、自分で言語を作るのと同じレベルなのです。そして、それが面倒な人のために用

意されているのが、BASIC であるといえるのです！

何だか、スゴクむずかしいことをやろうとしているように思えるかもしれませんが、BASIC のように使用目的がハッキリしていない言語でプログラムを作ろう、というわけではありません。これから作るマシン語プログラムは、自分のゲームにだけ通用し、しかも使用上の制限は勝手につけていいのですから、いたって気楽なものです。この使用目的を限定するというのが、結局は処理速度を早くできることにつながっていくわけです。そこで、まずはゲームの顔ともいえるグラフィック画面に対して、ゲームに便利のように制限をつけることにします

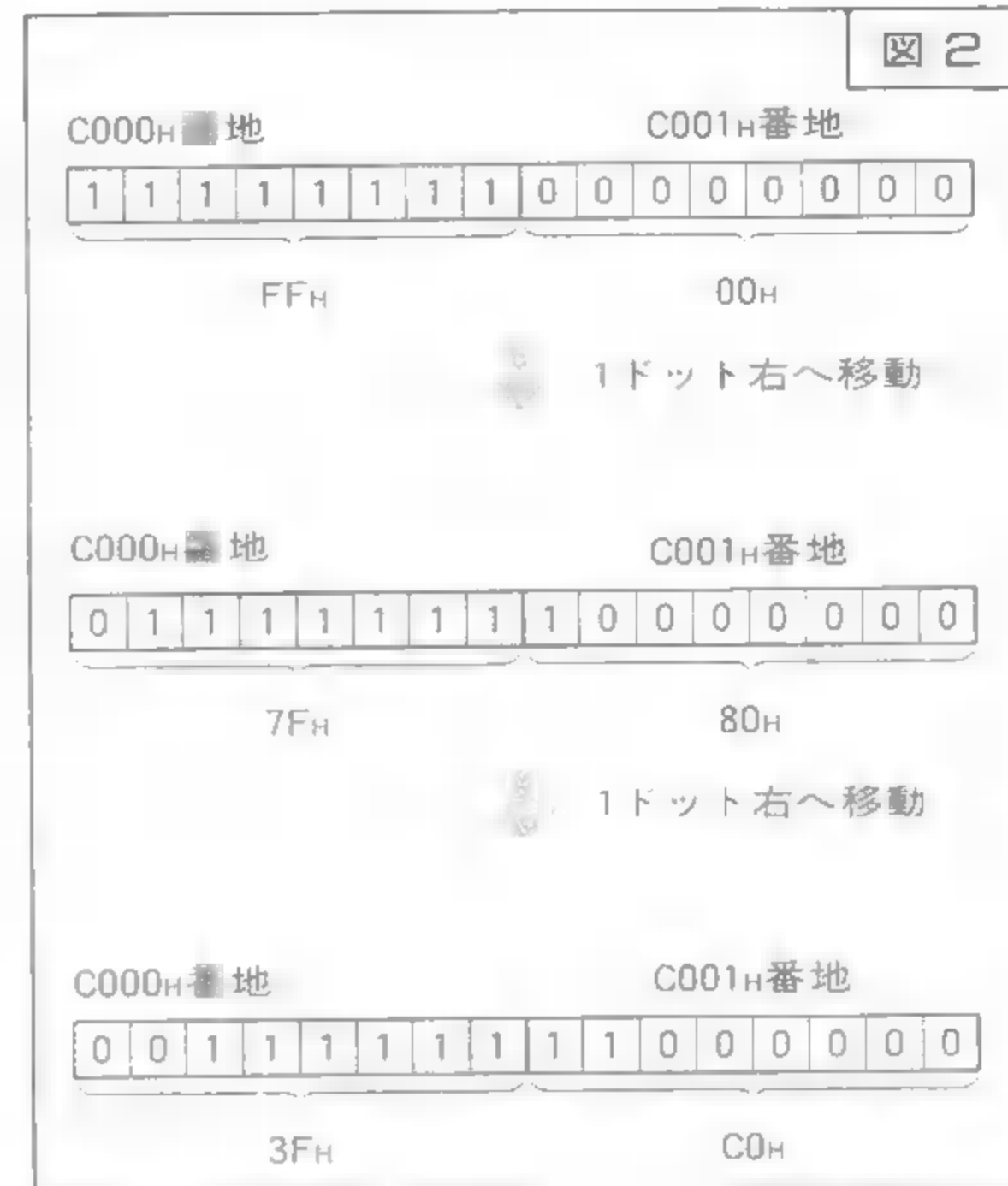
図 1



BASIC では、グラフィック画面を(0, 0)～(639, 199)という1ドット単位の座標によって管理していました。マシン語ゲームでは特殊なケースを除き、横8ドット、縦4ドットの正方形を1マスとして、グラフィック画面を管理します。これを本書ではゲーム座標といい、座標で表わすと(0, 0)～(79, 49)ということになります。図1をみてください。ゲーム・デザインを考える時にもこの座標を基準にして作成することになりますので、最初に用意した方眼紙にこの座標を書き込んでおくとう便利です。

なぜ、1ドット単位で管理しないかという、最大の理由は面倒だからです。面倒ということは、すなわち処理に時間がかかり過ぎるということです。1章で、グラフィック画面に短い白線を描きましたが、これを右に1ドットだけずらすということになると図2のように、データを入れるグラフィック・V-RAMのアドレスは2バイトに渡ってしまい、データも2つ用意しなければなりません。次にもう1ドット右にずらすとなると、また別の2つのデータが必要になります。

データは計算により求めることもできますが、その分時間がかかります。さらに面倒なことには、すでに何か背景に色がある場合は、重ね合わせ(6章参照)と呼ばれる処理をしないと、白線だけでなく余分な黒線も描くことになってしまうのです。このような理由から、横は8ドット単位で処理するのが一番都合がいいのです。縦は別に1ドット単位でも問題はないのですが、縦横同じサイズの方が座標として扱いやすいため、4ドット毎にしてゲーム座標として



いるのです。

しかし、棒状のメーターを増減させたい時などは、1ドットか2ドット単位の変化でないとメーターらしくなりませんから、そのような時は面倒でも重ね合わせの処理をしながら表示させなければなりません。

次に、表示するパターンのサイズですが、32×16ドットまたは24×12ドットとするのが普通です。これは、画面のサイズから判断して、あまり大きいパターンではゲーム・デザインがたいへんだし、マシン語といえども表示するのに時間がかかり過ぎるからです。そのため、本書では文字や数字以外のキャラクタ・パターンは、表示ルーチンのプログラムも含めて32×16ドットを基準にしています。これは、ゲーム座標でいうと4×4コマに相当しており、パソコン・ゲームにおいては最も標準的なサイズのキャラクタ・パターンとなっています。

2. 豆腐…とりあえず白い四角形を表示

ごく常識的に考えれば、コンピュータと出会って最初に目を通すのは、BASIC についてのマニュアルや参考書でしょう。そして、『PRINT 1+1』などとコンピュータをバカにしたような計算をさせてみて、その当たり前の結果に「フム、フム!!」とうなずきながら、満足するというのが一般的な入門光景です。その内に、グラフィック関係の命令を見つけ出してきて、画面のアチコちに線を引いたり、四角形や円を描きながら、「シメシメ、これで絵が描けるゾ…!!」なんて思いながら、コンピュータにのめり込んでいくわけです。

マシン語を覚える際にも、このように視覚に訴えながら進んでいくと、理解する楽しさが増してきます。プログラムを追うだ

けでは、どうしても面白さ、わかりやすさという点で不満が残ってしまうものです。ちょうど、小説よりもマンガの方が、情景がハッキリするのと同じようなことです。頭の中だけで理解するより、視覚に訴えて理解する方が間違いも少ないし、進歩の度合いも速いといえるでしょう。

パターン・サイズと座標の取り方が決まったところで、画面の任意の位置に 32×16 ドット 白い正方形を表示するプログラムを作成してみましょう。

List 2-1 の左側に、ソースを作成する時の行番号が出ています。行番号は、自由につけてもかまわないのですが、できるだけ同じ番号にした方が、後で間違いをチェックしやすくなります。また、ラベルというものはプログラムを作った本人以外には、なかなか理解しにくいものなので、省略前のものも載せてあります。ただし、英文法は無知していますので、そのつもりで見てください。なお、1 章でも書きましたが、今後ニーマニックでわからない命令があった場合は、Appendix 4 のマシン語命令小辞典を見ながら理解してもらうことを前提としています。本文では命令そのものについての説明は避け、重要な語句やプログラムの概略を中心に説明をしてあります。では、まず白い真四角な豆腐ができるまでの工程を示した図 3 を見てください。

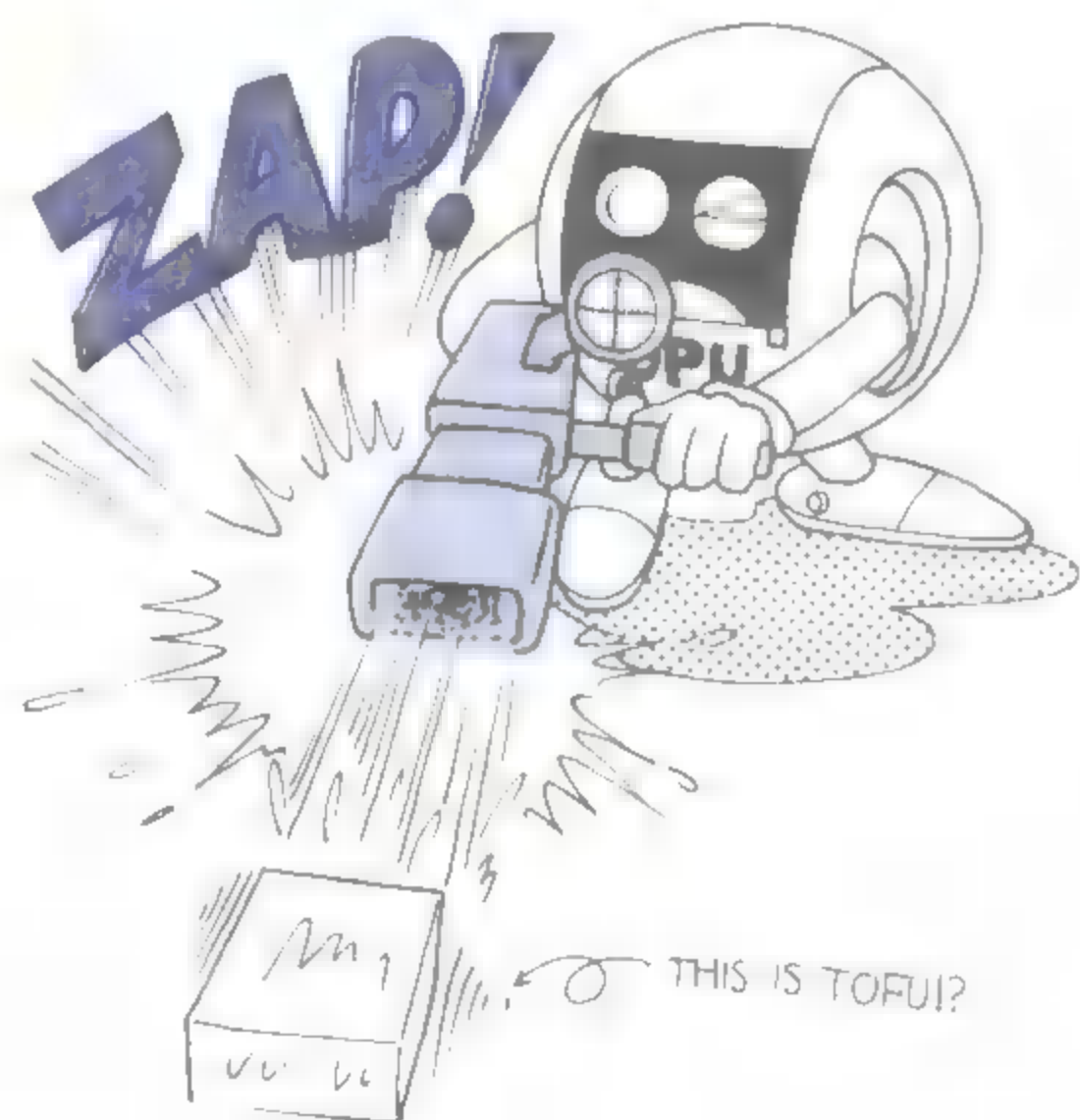
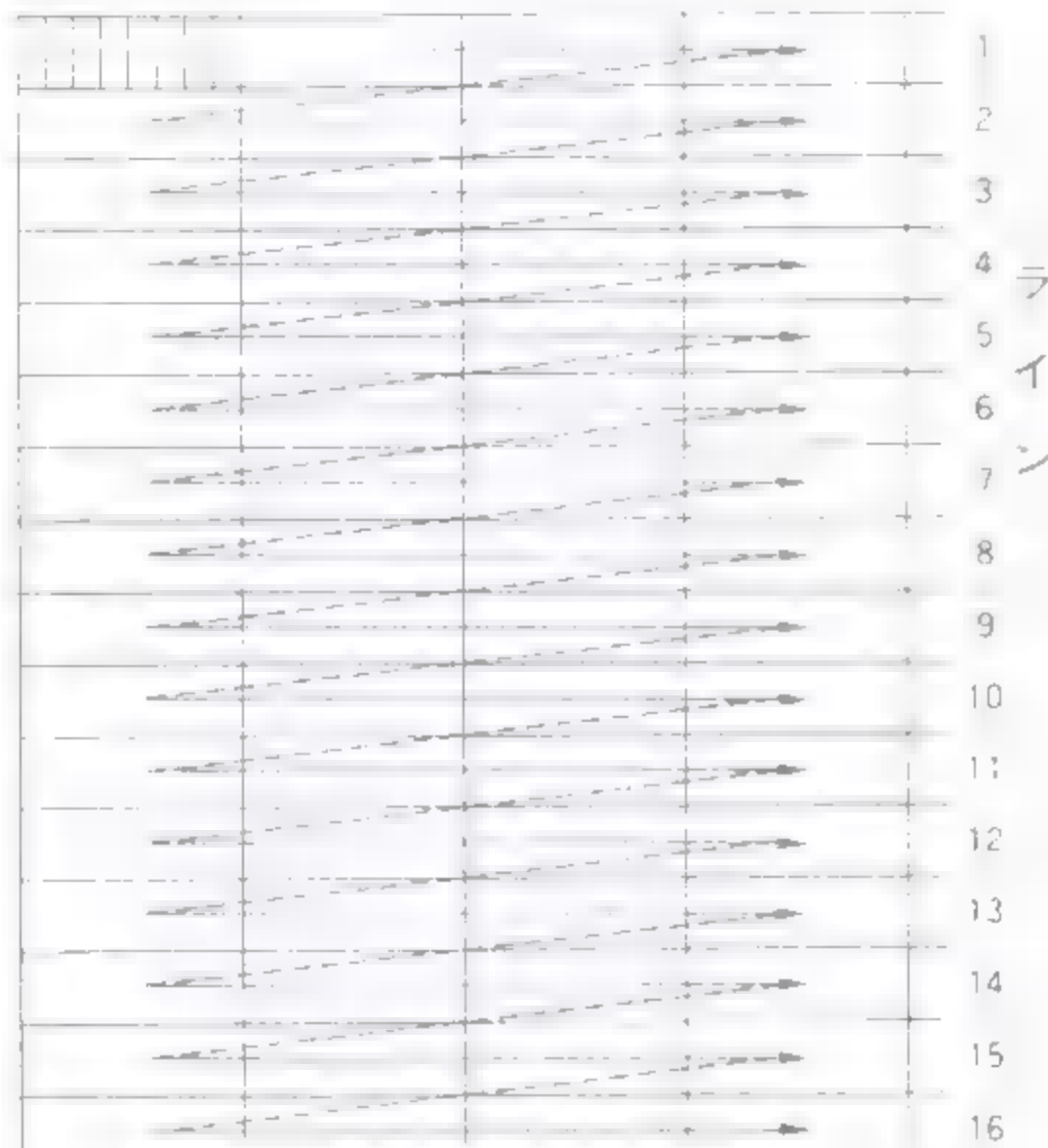


図 3

- ① 豆腐の表示アドレスを求める
- ② ブルー面に32×16ドットの正方形を描く

表示アドレス



(表示アドレスから、---に沿ってFFHを入れていく)

- ③ レッド面に32×16ドットの正方形を描く
(②と同様)
- ④ グリーン面に32×16ドットの正方形を描く
(②と同様)

このプログラムは、大きく分けると5つのブロックからできています。10000～10080行は、スタック・ポインタやV-RAMの先頭アドレスなどに名前をつけています。10090～10190行のTOUFUルーチンは、BCレジスタで示される位置を後述するXYADRルーチンで実際に四角形を表示するアドレスに変換した後、ブルー、レッド、グリーンの3画面に四角形を表示します。10380～10460行のXYADRルーチンは、BCレジスタで示されるゲーム座標をグラフィックV-RAMの実アドレスに変換してHLレジスタに入れます。10500～

10560行のTESTルーチンは、BCレジスタに表示する位置を入れてTOUFUルーチンをコールします。

なお、本書のプログラムは、すべてこのTEST(メイン・ルーチン)から実行するようにしています。そこで、プログラムを読む時には、まずTESTルーチンから読み始めるとプログラム全体の構成が理解しやすいでしょう。

また、1章のLD命令についての説明にあったように、BOXとXYADRの部分で早速LD命令の中で計算をさせています。BOXの方の例では、10進数と10進数の計算ですが、XYADRの方は16進数と10進数の計算になっていますね。このように、足し算、引き算であれば、10進数・16進数は問わずにアセンブラの方で計算してくれますので、こちらの手間が省けます。これも、アセンブラの便利な機能の1つです。

さて、豆腐の作り方がわかったところで、今回のプログラムで一番理解しにくい部分、SP(スタック・ポインタ)という言葉の意味について説明をしなければなりません。このスタック・ポインタについては、BASICからマシン語ルーチンに入る際にも関係のあるたいへん重要な部分ですので、ここはひとつ腰を据えてジックリと読むようにしてください。どちらかというと、メインの豆腐作成ルーチンを理解するよりも大切であり、ここを軽視すると将来思わぬ落とし穴に陥ることになります。

まずはメインのルーチンの中で、CALL命令とPUSH、POP命令が、どのような役割で使われているのかを調べてみましょう。CALL～RET命令は、BASICでいうと

GOSUB～RETURN 命令と同じようなものです。それに対して、PUSH, POP 命令というのは BASIC にはない考え方で、一時的にレジスタの値を保存しておき、必要な時に出すというものです。これは、BASIC のように変数を自由に取れないマシン語では、非常に便利な存在となっています。

次に、CPU の動きに目を向けてみます。CPU はメモリにある命令を実行する前に、まず次の命令がある番地をプログラム・カウンタに記録をしてから、命令の実行に移ります。命令の実行が終わると、再びプログラム・カウンタにある番地の命令を読み、また次の命令のある番地をプログラム・カウンタに記録する…ということを繰り返し行なっているのです。結構、手間のかかることをしていますね。しかし、これだけでは呼ばれた先で RET 命令に出会っても、元の流れに戻ることはできません。きちんと戻るためには、CALL 命令があった場所でプログラム・カウンタとは別に、CALL 命令の次の命令のある番地を、RET の戻り先として、どこかに記録しておかなければならないはずで

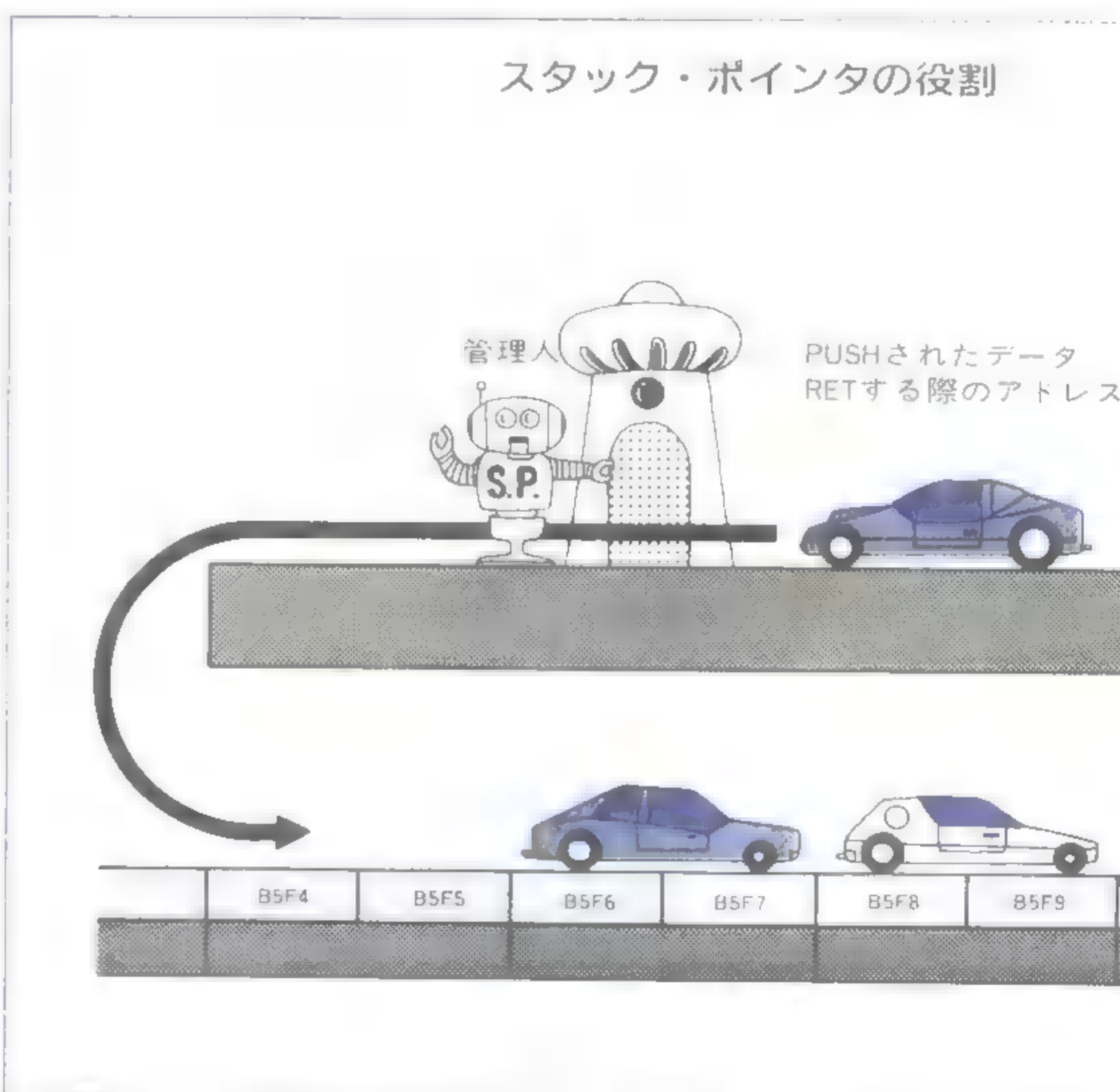
一方、PUSH 命令は一時的にペアレジスタの値を保存するといいますが、一体どこに保存しているのでしょうか。CALL 命令にしても PUSH 命令にしても、プログラムによって使用される回数が違うので、そのための記録エリアをどのくらい用意すればいいのかまったく不明です。そこで、これらのデータを記録するために、メモリの一部を最初に記録エリアとして用意する必要があります。

このような特殊な記録エリアをスタック

・エリアといい、そこでこれらのデータの入ったメモリの番地を記憶する特別なレジスタをスタック・ポインタ (SP) というのです。ですから、PUSH 命令や CALL 命令を使う場合には、最初に SP を設定してスタック・エリアを確保しないと、CPU が勝手にスタック・エリアを作り、必要なプログラムやデータを破壊したり、画面を乱したりする恐れがあります。

今回のプログラムのように B600H 番地を SP とした場合、実際のデータは図 4 のように、B5FFH 番地から番地の若い方へと、2 バイト単位で入っていきます。ここで、スタック・エリアを駐車場、データを駐車場へ入る車とみなし、駐車場には出入口が 1 ケ

スタック・ポインタの役割



所しかないと仮定します。管理人の SP は車をドンドン引き受けて、駐車場の一番奥から入れていきます。しかし、車を出す時は出口に近いものからしか出せませんから、SP は出口に一番近い車の駐車位置(番地)だけを常に覚えているのです。このように最後に入れたものを最初に出すというルールを、LAST IN FIRST OUT、または FIRST IN LAST OUT の原則といいます。

この管理人の SP は、またたいへんいい加減で、車を出しに来た者には、元の持ち主でなくても車を渡してしまうのです。例えば、HL から預かった車でも、DE が取りに来れば DE に、BC が取りに来れば BC に、という具合にいちいち確認などせずに渡してしまうのです。ですから、車の持ち

主(結局はプログラムを組むあなたのことですよ)が出し入れの順番を、シッカリ把握しなければならないのです。

この原則を踏まえた上で、CALL 命令や PUSH, POP 命令を使わないと、恐ろしい暴走に出会うことになります。しかし、実際には1つの CALL ルーチンの中で、PUSH と POP の使用回数が同じであって、スタック・エリアとして 100 バイト位のメモリを確保してあれば、特に問題は起きないものです。スタック・エリアとはこのように重要な部分であるだけに、その設定に関しては次のような注意すべき点があります。

1. C000H 番地より以前に設定する。

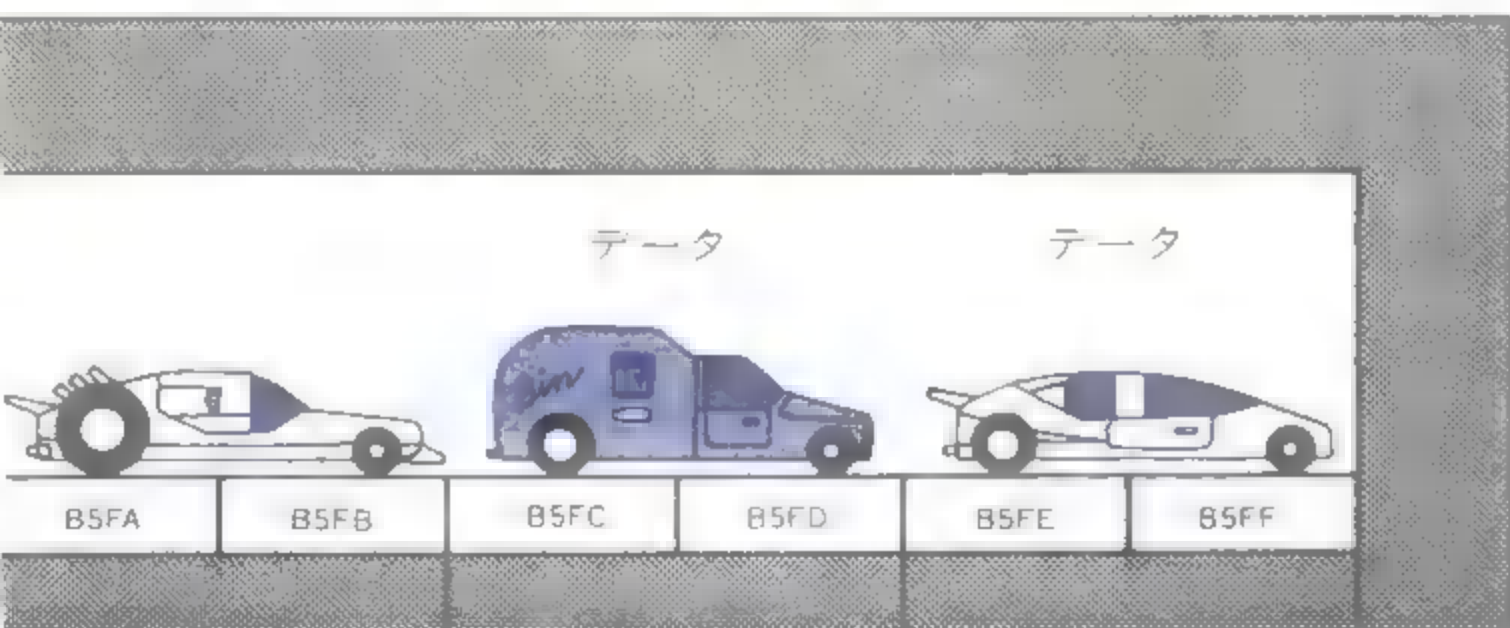
これは、グラフィック・V-RAM にバンク切り換えをしても、スタックの中身が切り換わらないようにするためです。同じような理由から、バンク切り換えも C000H 番地以前で実行しないといけませんので、念のため…。

2. BASIC の CLEAR 文によってマシン語エリアを確保し、そのエリア内だけでプログラムを組む場合は、スタック・ポインタを設定しなくてもマシン語プログラムを破壊することはない。

CLEAR 文を宣言すると、宣言した次の番地から E5FFH 番地までが完全にフリーエリアとなり、マシン語プログラムが BASIC の変数などによって破壊されることはなくなります。そして、このフリーエリア外に 16 バイトのスタック・エリアが自動的に確保されます。したがって、スタックの使用が 16 バイト(8 個分)以下であり、マシン語

図 4

- データの入庫はいくらでも引き受ける
- すぐ出せるデータの先頭アドレスだけを記憶している
- 出せと言われれば相手が違っても出す



プログラムが確保されたエリアから出ない時には、スタック・ポインタの設定は必要ありません。また、BASIC からマシン語プログラムに入り、再びBASICに戻る場合には、入った時のスタック・ポインタになっていなければなりません。つまり、簡単なマシン語プログラムをBASICのサブルーチンとして使う場合は、スタック・ポインタの設定はしない方が安全といえるわけです。当然のことですが、マシン語ルーチンの中でグラフィックV-RAMにバンク切り換えをする場合は、C000H番地以前でCLEAR文を宣言しなければなりません。なお、マシン語とCLEAR文とは切っても切れない関係にありますので、マニュアルを良く読んで理解することが大切です。

要するに、スタック・ポインタの設定を機械にまかせるか自分で管理するかは、そのマシン語プログラムの内容にかかっているといえるのです。本書では、CLEAR文によるマシン語エリアの確保は必要としますが、最終的な目標を本格的なオールマシン語ゲームということにしていますので、テスト・プログラムも含めてすべてスタック・

ポインタを設定して、キチンと管理するようにしています。しかし、マシン語プログラムをBASICプログラムのサブルーチンとして使う場合は、前述の注意を守った上でスタック・ポインタの管理を行なわないと、BASICにうまく戻れなくなったり、BASICの変数を破壊したりする可能性がありますので、くれぐれも間違いのないようにしてください。

さて、このプログラムでゲーム座標から、実際のグラフィック・V-RAMのアドレスに変換しているXYADRという部分ですが、ここでの計算式は次の通りです。

求めるアドレス

$$= \text{BEC0H} + 140\text{H} \times (\text{B} + 1) + \text{C}$$

では、テストの実行です。テストのスタート・アドレスはすべてD000H番地となっており、これは5章までは変わりません。せっかくですから、BCレジスタ(表示座標)の値を変えて、色々な場所に豆腐を出してみてください。アセンブル後モニタから、

h]GD000 []

としてみてください…。アッという間に「豆腐のイッチョ上がり」となりましたね。

List 2-1 豆腐の表示

```

10000 ;***** List 2-1 *****
      ;
      B600 STACK: EQU 0B600H ;STACK pointer
      C000 VTOP: EQU 0C000H ;V-ram TOP address
      0050 HLEN: EQU 80 ;Horizontal LENgth —横のバイト
      0140 HLEN4: EQU 320 ;HLEN x 4 総数
10060 ;

```



```

                                ORG 0BE00H
;
;TOUFU: ;TOUFU
BE00 CD29BE CALL XYADR
BE03 3EFF LD A,0FFH
BE05 D35C OUT (5CH),A
BE07 CD17BE CALL BOX
BE0A D35D OUT (5DH),A
BE0C CD17BE CALL BOX
BE0F D35E OUT (5EH),A
BE11 CD17BE CALL BOX
BE14 D35F OUT (5FH),A
BE16 C9 RET

;
;BOX: ;BOX
BE17 E5 PUSH HL
BE18 114D00 LD DE,HLEN-3
BE1B 0610 LD B,10H
BE1D LOOP: ;LOOP
BE1D 77 LD (HL),A
BE1E 23 INC HL
BE1F 77 LD (HL),A
BE20 23 INC HL
BE21 77 LD (HL),A
BE22 23 INC HL
BE23 77 LD (HL),A
BE24 19 ADD HL,DE
BE25 10F6 DJNZ LOOP
BE27 E1 POP HL
BE28 C9 RET

;
;XYADR: ;xy to Address
BE29 21C0BE LD HL,VTOP-HLEN4
BE2C 114001 LD DE,HLEN4
BE2F 04 INC B
BE30 XYLP: ;xy Loop
BE30 19 ADD HL,DE
BE31 10FD DJNZ XYLP
BE33 09 ADD HL,BC
BE34 C9 RET

;
                                ORG 0D000H
;
;TEST: ;TEST
D000 F3 DI
D001 3100B6 LD SP,STACK
D004 010000 LD BC,0000
D007 CD00BE CALL TOUFU
D00A FB EI
D00B FF RST 38H

```

の四角形を表示するルーチン

↓ 遊

3. パターン…キャラクタの作成

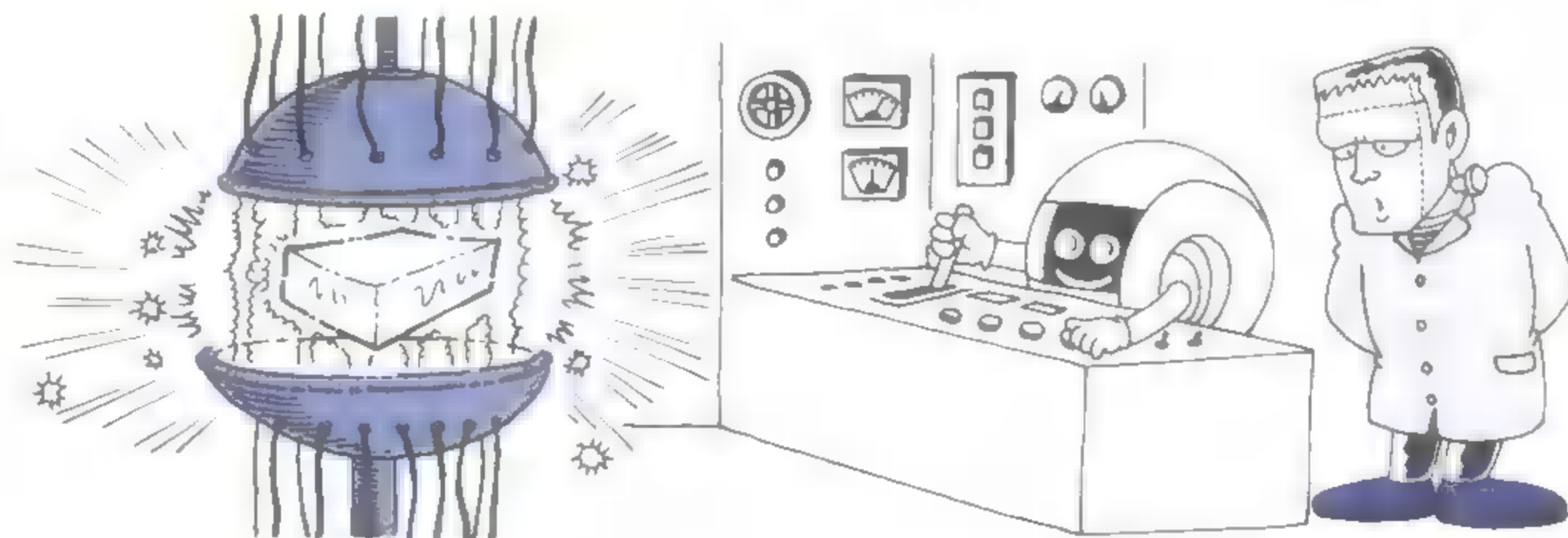
豆腐作りの修行は、いかがでしたか。画面の中のどこにでも、豆腐を作れるようになれば、もう一流の豆腐職人です。ここで、このルーチンの先頭についている TOUFU というラベル名、これを1つの言葉と考えるとうどうでしょうか。これは、すでにアルファベットの文字ではなく、オリジナルな言語であるといえます。それが証拠に、BASIC には TOUFU などという命令は、どこを捜してもありません。サイズは一定、表示位置はゲーム座標による、というような制限はありますが、そういう言葉なのですからそれでいいのです。自分で作った、自分のためだけの言葉ですから、他人が使うことなど考える必要もないのです。こんなところが、マシン語のたまらない魅力であり、一方とっつきにくくしていた原因でもあったのです。しかし、豆腐一丁でその壁はもろくも崩れたことでしょう。「豆腐の角に頭をぶつけて、死んでしまえ／＼」というのは、この壁に対する格言

だったのですネ？

これから、豆腐を卒業して実際にパターンを画面に表示する段階に入るわけですが、パターンを表示するにはまずそのデータがなければなりません。ここでは、パターン・データ作成の方法として、パターン・エディタを実際に使いながら、次節で使うデータを作成することにしましょう。

まずデータの作成方法ですが、方眼紙にドットで絵を描いて、それを手作業で16進数に直す…なんていうことを考えた方はいないと思いますが、現実はそのと同じことをコンピュータにさせて作るのです。Appendix 3 の pated が、そのためのプログラムですが、ここでテスト的に利用するだけでなく、将来も使えるように色々と便利な機能をつけてあります。マシン語の勉強とは少し離れるかもしれませんが、これなくしてはパターン表示のテストもできませんから、頑張って打ち込んでください。

リストを打ち終えたら、走らせる前にか

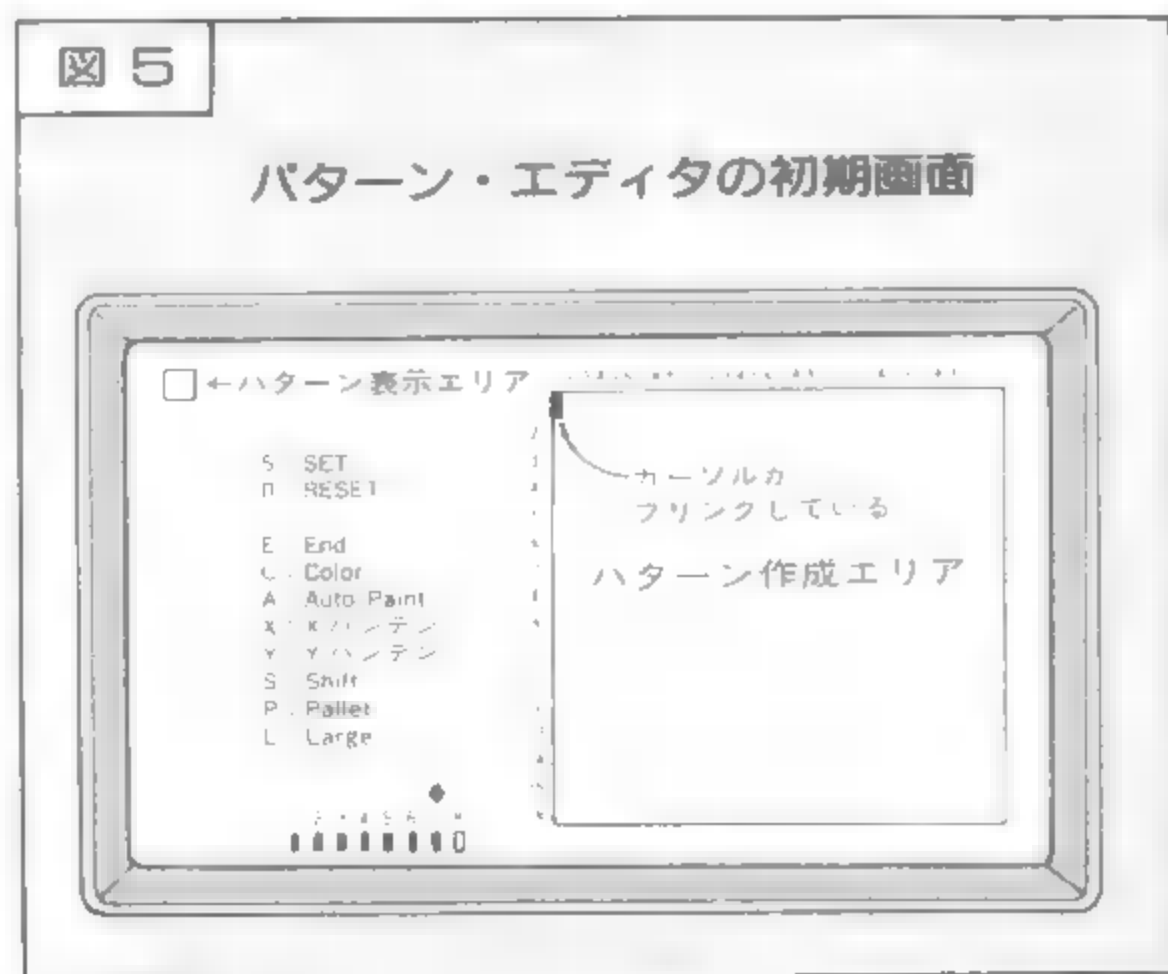


ならずセーブするのは、もう常識ですね。それから、DISK-BASICの場合このプログラムは、How many files? に対し0を入力しないとメモリ不足でエラーになりますので、実行する際には忘れないようにしてください。では、走らせてみましょう。画面に次のようなメッセージがでます。

パターン サイズ (Max,X=56,Max,Y=24)
X(DOT), Y(DOT)?

これは、これから作るパターンの大きさを聞いているのです。横(最大56)と縦(最大24)のドット数を、例えば「32,16」のように一度に入力してください。横は8の倍数でなくても受け付けますが、データは結局8ドット単位で作られるので、最初から8の倍数で入れる方がいいと思います。ここでは「32,16」とします。

画面にあなたが答えたサイズの大きさの四角形ができ、その中でカーソルが点滅しています。左下には、カラー・パターンがあり、パレットコード7の上に●があります。



す。これは、現在セットされている色を示しているのですが、このパターンの中でパレットコード8というおかしなものがあります。もちろん、現実こんな色があるわけはありません。この特殊な色は、重ね合わせ用のデータ作成をする時にだけ使い、透明(背景となる)を意味するものなのです。ただし、データがあっても重ね合わせ表示プログラムがないと、何の役にも立ちませんので、6章までは関係のない色(?)といえます。なお、作成されたパターン上では黒で表示されますので、黒の代用として使ってもかまいません。カーソル移動とその他の機能は次ページの表1の通りです。

カラーページ④のキャラクタ・パターンの中から好きなものを1つ作成してみましょう。少しでもオリジナル性を出すために、色を自分の好みで変えるのも一案です。

完成したらE(エンド)コマンドでデータをメモリに落とし、忘れずにセーブしておきます。データは1(B..., R..., G...), 2(B·R·G, B·R·G, B·R·G,...), 3(透明·B·R·G, 透明·B·R·G, 透明·B·R·G, ...)の3種類があり、更に必要に応じてデータの並びを変えたり、削除したりできるようになっています。(B: Blue, R: Red, G: Green)特に指示のない場合は、1のタイプのデータを選び、その後の「Change Data./」の表示にはそのまま[Enter]を押してください。今回は32, 16ドットのサイズにしていたから、データ・アドレスがB500H-B5BFHと表示されたはずですが、グラフィック各面のデータ数は同じですから、このデータ・アドレスの内訳は次のようになります。

B500H-B53FH 番地

…ブルー面のグラフィック・データ

B540H-B57FH 番地

…レッド面のグラフィック・データ

B580H-B5BFH 番地

…グリーン面のグラフィック・データ

作ったパターンのデータは、カラーページの前の「本書のキャラクタの作り方」を参考にしてセーブしておいてください。

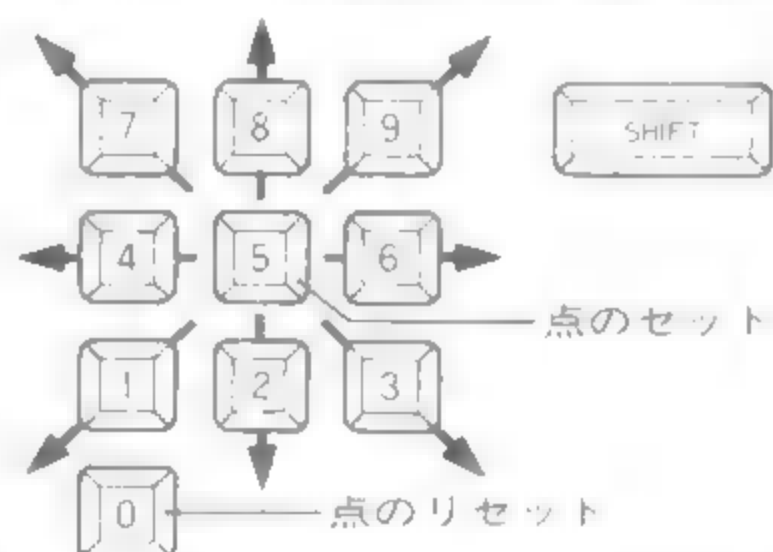
なお、このプログラムでデータ作成に使用しているマシン語部分については、プログラム中に DATA 文の形で挿入されていますので、実行は RUN で OK です。

また BASIC プログラムの中で、4280 番地のマシン語ルーチンが何度も使われていますが、これは BASIC ROM にあるカーソルの表示を行なうルーチンです。BASIC は命令によっては、カーソルを出したくても出せないものがあります。そういう時に、このルーチンを呼んでいるのです。

表 1

パターンエディタの■

カーソルの移動および点のセット/リセット



B ……ブルー面
R ……レッド面
G ……グリーン面
T ……透明

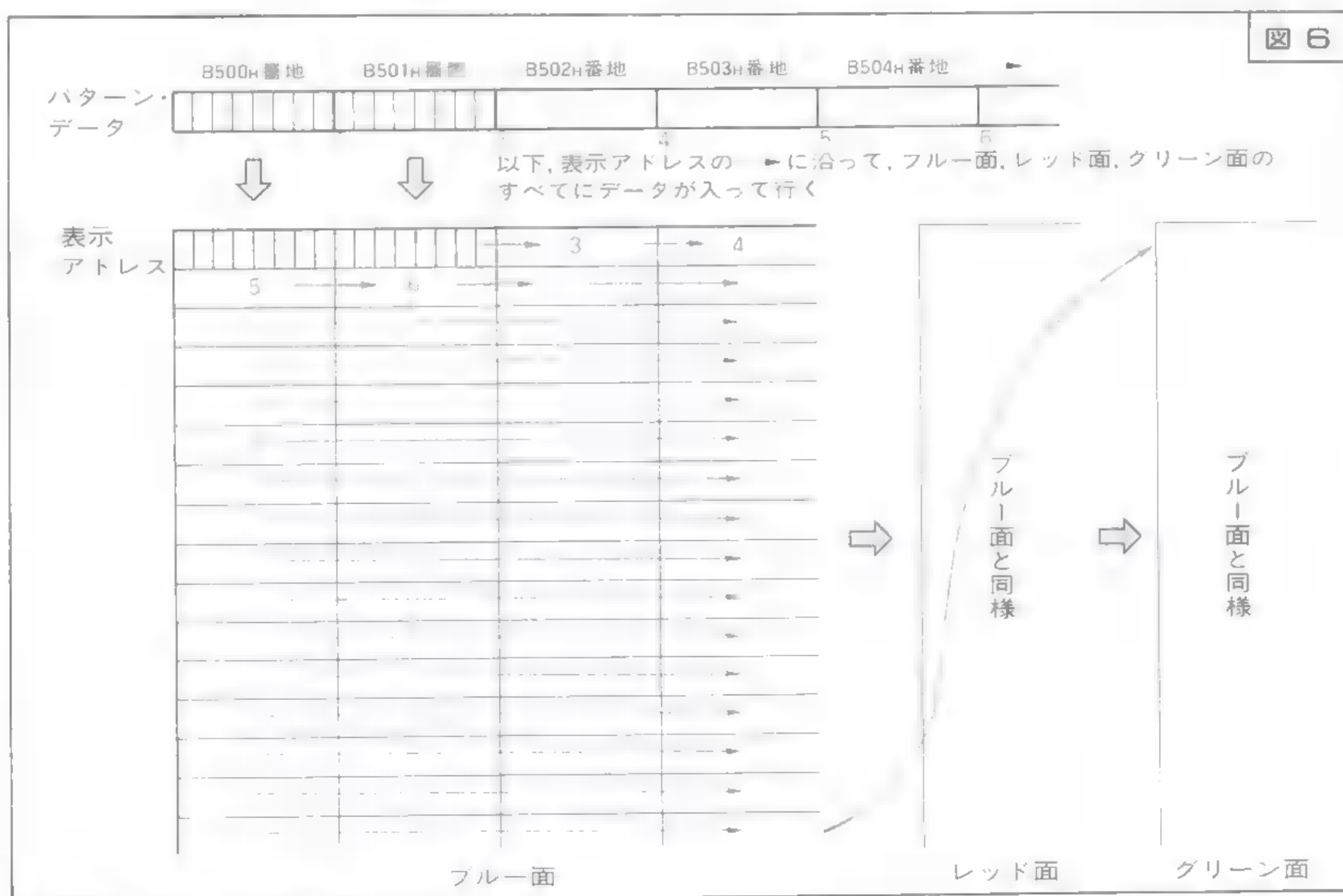
プログラムの終了		パターン・データを作成し B500H 番地からメモリに格納する 1. データ・タイプの選択(3種類のデータ・タイプがある) 2. チェンジ・データ (変更: データ番号を押し, 変更するバンク[R], [G], [B]を入力) (削除: データ番号を押し, [0]または[END]を入力)
セットする点の色を選択		0~7のパレットコードを入力。透明を示す8は、データ・タイプ3のTBRG, TBRG, …以外では0(黒)とみなされる
ペイント		パターンのペイント 全部を塗りつぶすモード[A]と一部を塗るモード[P]がある
X軸方向の反転		パターンの左右を入れ換える
Y軸方向の反転		パターンの上下を入れ換える
パターンのシフト		はじめの方向を[2], [4], [6], [8]のキーで入力。次に移動ドット数を入力
パレット変更		パレットを変更する
ラージ機能		パターン表示エリア(C000H番地)に表示されているパターンをパターン作成エリアに取り込む

4. パターン表示…キャラクタ登場

1つのゲームを作る時に、パターン数は一体いくつくらい必要になるかという、これが千差万別なのです。文字や数字を除いたパターンが、少ないものでは20~30程度のゲームもあれば、200以上のパターンを使用しているものもあります。パターン数を多くすれば、それだけ動きがなめらかになりますが、メモリの効率や作る労力のことを考えると、一概に多ければいいともいえません。それよりも、少ないパターンデータで多く見せるということの方が、必要なのかもしれない。いずれにしても、ゲームを作るということは、パターン作成

という地味な作業も、避けては通れない部分だということです。

テスト用とはいえ、あなた自身で作ったパターン(キャラクタ)のデータが揃いました。これで本当に豆腐とオサラバできることになったわけです。しかし、パターンを表示するプログラム List 2-2 は、プログラマ的にはグラフィック・V-RAM にこれまで入れていた FFH を、パターンに置き換えるだけです。List 2-1 とほとんど変わりがないといえます。スクリーン・エディットによって、プログラムの変更も実に簡単に済んでしまったと思います。



[illegible]

然ですし、レジスタペアとしての役目はできそうにありません。まるで単独では困るような時だけ一緒になっているようなものですね。それでは、フラグ・レジスタとはどんなレジスタかというと、実は数値を代入するこれまでのレジスタと違い、足し算、引き算、論理演算、比較などの各種演算をした結果によって、ある決まった反応を示す特殊なレジスタなのです。このレジスタの内容を表2に示しておきます。

フラグ・レジスタの役目は、演算に対しビット単位で1か0を示すということです。そして、フラグの場合はビットが1になっている所を《フラグが立っている》といます。特にゼロフラグの場合は、「ゼロの時には1になる」などと覚えようとすると混乱しますから、《ゼロになったらゼロフラグが立つ》と単純に言葉で覚えた方がハッキリします。

これらのフラグの中で、よく使われるのはゼロフラグとキャリーフラグの2つで、その他はほとんど使わずに済みます。もちろん、マシン語に慣れてきたならば使うだけの価値はあるのですが、ここで無理に覚えるほどのことではありません。本書にあるプログラムも、ほとんどがこの2つのフラグだけで処理されています。

だから、あなたは……

《ゼロになったらゼロフラグが立つ》

《最上位ビットを越えた桁上げ、桁借りがあったらキャリーフラグが立つ》

……ということだけを、今は覚えればいいのです。

フラグの存在が確認できたところで、こ

れをどのように利用するかということですが、先ほどの例のように条件分岐としてよく使用されます(キャリーフラグは、算術演算、ローテート、シフト命令でも用いる)。具体的には、

JR

ゼロフラグ、キャリーフラグによる分岐が可能

JP

サイン、ゼロ、……、キャリーの各フラグによる分岐が可能

CALL

サイン、ゼロ、……、キャリーの各フラグによる分岐が可能

RET

サイン、ゼロ、……、キャリーの各フラグによる分岐が可能

との組み合わせで使用されることになります。先ほどの例が、なぜCレジスタの値を見てジャンプしているのではないのか、次のようにすると明確になります。

```
DEC C ; C=C-1
```

```
LD C, 0 ;
```

C=0とする…LD命令ではフラグは変化はない

```
JR NZ, LOOP1 ;
```

ゼロフラグが立っていなければ LOOP1へジャンプ

プログラムの的には、まったく意味がなくなってしまうかもしれませんが、それでも DEC Cをした時点でCの値が0でなければ、LOOP1へジャンプすることになります。このように、命令によってフラグは影響を受けたり受けなかったりしますので、条件分岐をする際には注意が必要です。特に、アキュムレータの値を最後に戻す場合など、POP AFではフラグも変化することになりますので、間違えないようにしなければなりません。命令とフラグ変化の関係については、

Appendix 2 のマシン語インストラクション一覧表のフラグの項目を見れば、表 3 のような形で示されています。

さて、フラグが理解できたところで、このパターン表示プログラムのテストをしてみましょう。先ほど作成したパターン・データを、B500H 番地に再ロードします。BC レジスタの値を変えて実行することにより、画面の好みの位置にオリジナル・パターンを表示できるようになったはずですが…？

このプログラムでも、簡単なゲームであれば特に問題はありませんが、まだまだこれは本格的なパターン表示ルーチンとはいえません。その理由は速度です。実際のマシン語ゲームでは、7～8 割がパターンを表示するための時間に費やされています。ですから、パターン表示ルーチンをできるだけ高速にすることが、すなわちゲームの高速化につながるのです。このことは、ゲーム中に表示できるパターン数にユトリがあれば、その分作れるゲームにも幅が出てくるということを意味しています。そこで、レジスタの利用法やアドレスの計算方法を全く変えて、速度だけを追及したプログラムが次の List 2-3 です。そして、これから我

々が使っていくのも、当然こちらの高速表示の方です。

まず、グラフィック・V-RAM のアドレス計算方法に工夫を凝らしています。ゲーム座標での Y 軸の +1 はアドレス上は +140H (10 進数では +320 バイト) になっていますが、この増加分である 320 を 64 と 256 に分解します。そして、計算式を下のようになり変えます。式そのものは複雑には見えませんが、プログラム上はループを使わなくて済むために、速度のアップと位置による処理時間のバラツキが消えるという利点が生まれるのです。

求めるアドレス

$$\begin{aligned} &= C000H + B \times (64 + 256) + C \\ &= C000H + B \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 + B \times 100H + C \\ &= C000H + B \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 + BC \end{aligned}$$

これまで座標を表わすために単独で使われていた BC レジスタを、そのままレジスタペアとすることにより、 $B \times 100H + C = BC$ を実現しています。

また、このプログラムではパターン・データをダイレクトにアドレスで指示するのではなく、現実のゲームにそくして、パターン数が増えてもポインタとなるデータ・アドレスを追加するだけで済むので、パターンの管理が楽になるのです。パターン番号は 0～FFH まで取れますから、不足することはまずありません。先ほどのパターンは No.1 としましたので、PDBASE で示されているように、データ・アドレスは B6C0H 番地からになります。そこで、次のように転送してから再セーブしてください。

表 3	フラグ変化の略号
.	フラグの変化はない
1	かならずフラグが立つ
0	かならずフラグがリセットされる
↓	フラグの変化は演算結果による

MON 

h]MB500, B5BF, B6CO 

h]


セーブ・アドレス=B6CO_H番地~B77F_H番地

さて、表示プログラムの内容ですが、具体的な方法は図6とまったく同じものです。違いは、使用する命令だけですが、ここでは速度追求のためレジスタの数が足りなくなり、スタック・ポインタをも単なるレジスタペアとして使っています。そのため、その間はPUSH, POPやCALL命令が使えないのはもちろんのこと、BOXルーチンの最後にはまたスタック・ポインタを元の値に戻さなければなりません。そこで、最後にスタック・ポインタの値を設定する部分に、BOXルーチンの最初で、直接スタック・ポインタの値を書いているのです。また、

CレジスタにFFHを入れているのは、LDI命令によってBCレジスタの値が-1されても、Bレジスタの値が変化しないようにするためです。

テストの実行は、同じくD000H番地からですが、ここでパターンを1つ表示させるくらいでは速度の差はほとんどかわらないと思います。しかし、実際には倍くらいの速度で表示されていますので、パターンをたくさん表示するゲームでは、この差はたいへんな違いとなって表われてくるのです。

これから、3章にかけて小さなサブルーチンを、テストによって確認しながら、1つの大きなプログラムを構成していきますが、このList 2-3がその第1回目ということになります。そこで、次の点に注意しながら、今後各プログラムを作成していくようにしてください。

- (1) List 2-3以降のプログラムは、打ち込みしたいアスキー・セーブ「SAVE」ファイル名, A すること。次にList 2-3から、打ち込んだプログラムまでをBASICのMERGE命令を使いアペンドする。アペンドしたプログラムは、どの段階でもアセンブルすれば実行できる(実行はモニタからGD000 による)。
- (2) プログラムはグラフィック(G), ノン・グラフィック(N), テスト(T)の3つに分けて書かれており、それぞれ開始アドレスが違っている。プログラムが、どこに属するかは、各ルーチンの最初のコメント欄にG, N, Tの印で示してある。
- (3) 新たに作成するプログラムは、左側にある行番号通り打つこと。前のプログラムに追加する場合、まちがいなくアペンドできる。単独ではプログラムとして成立しないので、行番号も同じにして打ち込むこと
- (4) テスト・プログラムはすべて50000行から作るようになっている。打ち込む時には、前回のリストを利用してスクリーン・エディットしてもかまわないが、不要部分は必ずDELETEすること。
- (5) テスト・プログラムの実行に際しては、プログラムの他にパターン・データが必要になる場合がある。パターン・データは、プログラムのアセンブル後ロードすること。

List 2-2 パターンの表示

```

;***** List 2-2 *****
;
B600      STACK: EQU 0B600H ;STACK pointer
C000      VTOP:  EQU 0C000H ;V-ram TOP address
0050      HLEN:  EQU 80    ;Horizontal LENGTH
0140      HLEN4: EQU 320    ;HLEN
;
;          ORG 0BE00H
;
BE00      DISP: ;DISPlay
BE00      CD2EBE  CALL XYADR
BE03      1100B5  LD DE,0B500H
BE06      D35C    OUT (5CH),A
BE08      CD18BE  CALL BOX
BE0B      D35D    OUT (5DH),A
BE0D      CD18BE  CALL BOX
BE10      D35E    OUT (5EH),A
BE12      CD18BE  CALL BOX
BE15      D35F    OUT (5FH),A
BE17      C9      RET
;
BE18      BOX: ;BOX
BE18      2A3DBE  LD HL,(DISPAD)
BE1B      011004  LD BC,410H
BE1E      LOOP1: ;LOOP 1
BE1E      C5      PUSH BC
BE1F      LOOP2: ;LOOP 2
BE1F      1A      LD A,(DE)
BE20      77      LD (HL),A
BE21      13      INC DE
BE22      23      INC HL
BE23      10FA    DJNZ LOOP2
BE25      014C00  LD BC,HLEN-4
BE28      09      ADD HL,BC
BE29      C1      POP BC
BE2A      0D      DEC C
BE2B      20F1    JR NZ,LOOP1
BE2D      C9      RET
;
BE2E      XYADR: ;XY to Address
BE2E      21C0BE  LD HL,VTOP-HLEN4
BE31      114001  LD DE,HLEN4
BE34      04      INC B
BE35      XYLP: ;XY Loop
BE35      19      ADD HL,DE
BE36      10FD    DJNZ XYLP
BE38      09      ADD HL,BC
BE39      223DBE  LD (DISPAD),HL
BE3C      C9      RET
;
BE3D      DISPAD: ;DISPlay Address

```

(C,B)から表示アドレスを求め
 (DISPAD)に入れるルーチ
 * 同様
 表示アドレスが入るワークエ
 アを確保


```

BE3D          DS    2
;
;          ORG    0D000H          ;プログラム開始アドレス = D000H
;
D000          TEST: ;TEST
D000 F3        DI
D001 3100B6    LD    SP,STACK
D004 010000    LD    BC,0000
D007 CD00BE    CALL DISP
D00A FB        EI
D00B FF        RST    38H

```

——メインルーチン

List 2-3 パターンの表示(図2.3版)

```

;***** List 2-3-G *****
;
B600          STACK: EQU 0B600H ;STACK pointer
C000          VTOP:  EQU 0C000H ;Vram TOP address
0050          HLEN:  EQU 80      ;Horizontal LENGTH
;
;          ORG    0BE00H
;
BE00          DISP: ;DISP lay          ——B, R, G 各面にパターンを表示
BE00 CD00C0    CALL XYADR
BE03 CD12C0    CALL PDADR
BE06 D35C      OUT    (5CH),A
BE08 CD18BE    CALL BOX
BE0B D35D      OUT    (5DH),A
BE0D CD18BE    CALL BOX
BE10 D35E      OUT    (5EH),A
BE12 CD18BE    CALL BOX
BE15 D35F      OUT    (5FH),A
BE17 C9        RET

;
BE18          BOX: ;BOX
BE18 ED7334BE  LD     (LDSP+1),SP
BE1C 314C00    LD     SP,HLEN-4
BE1F ED5B37BE  LD     DE,(DISPAD)
BE23 01FF10    LD     BC,10FFH
BE26          LOOP: ;LOOP
BE26 EDA0      LDI
BE28 EDA0      LDI
BE2A EDA0      LDI
BE2C EDA0      LDI
BE2E EB        EX     DE,HL
BE2F 39        ADD    HL,SP
BE30 EB        EX     DE,HL
BE31 10F3      DJNZ  LOOP
BE33          LDSP: ;Load Stack Pointer
BE33 310000    LD     SP,0000
BE36 C9        RET

```

——B, R, G 各面にパターンを表示
XYADR: X座標・Y座標を指定する
PDADR: 表示アドレスを指定する
BOX: 表示領域の幅・高さを求めるため
LOOP: 面・レット面・クリア 面にパターンを表示
LDSP: M 面を切り換え

——パターンを表示
XYADR: スタックポインタ(DSP+1)に退避
PDADR: 表示アドレスの増加バイト数
BOX: 表示領域の幅・高さを求めるため
LOOP: 面・レット面・クリア 面にパターンを表示
LDSP: 退避したスタックポインタを元に戻す

BE37
BE37

```

;
DISPAD: ;DISPlay Address
        DS    2

```

表示アドレスが入るワークエリアを確保

```

;
;***** List 2-3-N *****
;

```

```

        ORG    0C000H

```

```

;
XYADR:  ;XY to Address

```

表示アドレスを求め
DISPAD に入れ替

```

C000      LD    L,B
C000 68    LD    H,0
C001 2600  ADD    HL,HL
C003 29    ADD    HL,HL
C004 29    ADD    HL,HL
C005 29    ADD    HL,HL
C006 29    ADD    HL,HL
C007 29    ADD    HL,HL
C008 29    ADD    HL,HL
C009 09    ADD    HL,BC
C00A 0100C0 LD    BC,VTOP
C00D 09    ADD    HL,BC
C00E 2237BE LD    (DISPAD),HL
C011 C9    RET

```

```

;
PDADR:  ;Pattern Data Address

```

アドレスをHに

```

C012      LD    H,0
C012 2600  LD    L,A
C014 6F    ADD    HL,HL
C015 29    LD    DE,PDBASE
C016 111FC0 ADD    HL,DE
C019 19    LD    A,(HL)
C01A 7E    INC    HL
C01B 23    LD    H,(HL)
C01C 66    LD    L,A
C01D 6F    RET
C01E C9

```

```

;
PDBASE: ;Pattern Data BASE address

```

```

C01F      DW    0B600H,0B6C0H
C01F 00B6C0B6
C023 80B740B8 DW    0B780H,0B840H
C023 80B740B8
C027 00B9C0B9 DW    0B900H,0B9C0H
C027 00B9C0B9

```

```

;
;***** List 2-3-T *****
;

```

```

        ORG    0D000H

```

```

;
TEST:

```

```

D000      DI
D000 F3    LD    SP,STACK
D001 3100B6 LD    BC,0000
D004 010000 LD    A,1
D007 3E01  CALL    DISP
D009 CD00BE EI
D00C FB    RST    38H
D00D FF

```


5. パターン消去…キャラクターを動かす前に

画面へのパターンの表示が自由にできるようになれば、次はそれを動かしたいと思うのが人間の心理というものです。心理学的にもそれが当然なのではないかと思いますが？

もう大分前のことですが、学生時代にキタナイ格好をして、ヨーロッパを放浪していたことがあります。その時、ベルギーのブリュッセルにある、有名な小便小僧の像を見ようと思い、地図を片手にその近くまで行ったのですが、どうしても見つけれませんでした。…実は、余りに小さかったため、何度もその前を往復していたのです…それで、通りすがりの町の人に聞いたのですが、小便小僧という言葉がわからなかったため、恥ずかしながら道の真ん中で、大胆にも小便小僧の真似をしたのです。ジェスチャーは世界共通の言葉です。彼は、「アー、ワカッタ、ワカッタ!!」というような顔をして、それなら道の反対側にあるというのです。「おかしいナ、地図が間違っていたのかナァ…」と思いながら行ってみると、何とそこは公衆便所だったのです。

地元の人にとっては、あんなもの取るに足りないものなのでしょう。そこに、彼が旅行者の心理が読めず、こちらは逆に彼の心理が読めなかった原因があったのかもしれませんが、しかし、本書はお互いにマシン語ゲーム製作を目指しているわけですから、そのようなギャップなどあるはずがありませんね。期待通り(?)に、パターンの移動へ進んで行きます。

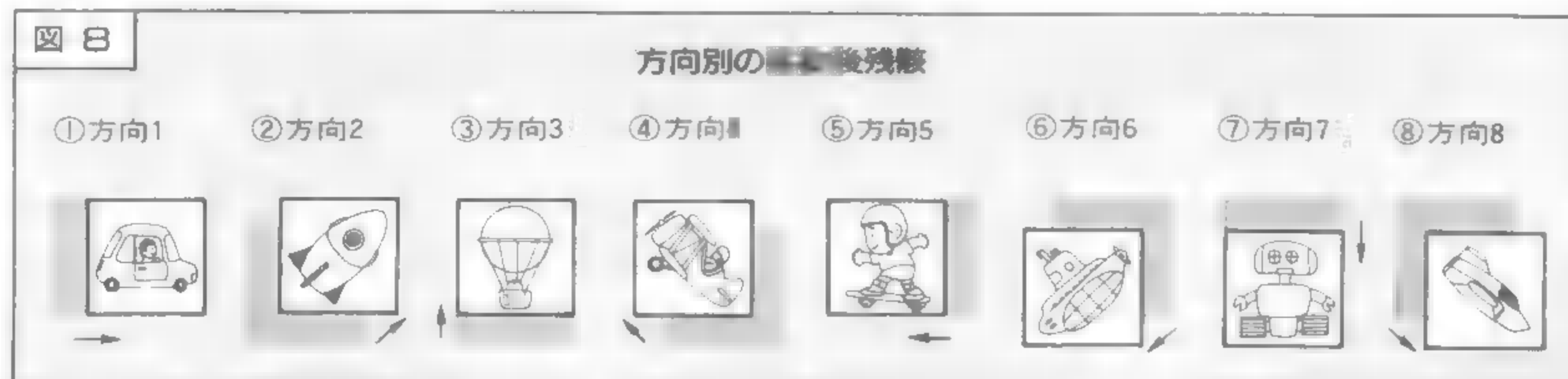
ここでは、パターンを動かすための準備として、画面上でモノが動くということはプログラ的にはどういう処理をすればいいのか、その原理と実際に次節で使うプログラムを作り、テストをしてみることにします。といっても、我々の管理をしているのはゲーム座標という横80コマ、縦50コマの小さな世界ですし、移動の単位もこの1コマを基準とすればいいのでそれほどむずかしいことはありません。

パターンの移動に際しては、どこに移動するのか、まず方向を数で決めなければなりません。そこで、ゲーム座標上で移動可能な方向すべてに、次のような方向番号をつけることにします。



これで、例えばゲーム座標で(10, 10)の位置にあるパターンを、方向1に1コマ動かす、というような表現ができるようになりました。この例では、移動後の座標は(11, 10)になりますが、それでは(11, 10)の位置

図 8



に新たにパターンを表示するだけでいいか
 というと、そうはいきませんね。前にあつ
 たパターンの残骸が、左側に少し残ってし
 まいます。本書ではパターンのサイズを 32
 ×16 ドットにしていますから、左、右上、
 下へ移動する際の、1 コマ移動後の残骸を
 見てみると上図のようになります。

結局、移動に際して邪魔になっているの
 はこの影の部分ですから、移動する前にこ
 の部分だけを消してしまえばいいというこ
 とです。そのためには、方向別の消去ルー
 チンを作らなければなりません。そこで、
 現在位置と移動方向を指示すれば、不要な
 部分を消去し、次座標を計算した上でその
 座標がゲームの画面からハズレるか否かも
 判定してくれるプログラムにすると、便利
 なものになります。少し長いかもしれませんが、List 2-4 を一通り読んでください。

List 2-4 の消去ルーチン(CLPCTX)は、
 List 2-1 豆腐作りルーチンでグラフィック
 ・V-RAM に入れていた FFH を 0 に変えた
 だけのことです。ただ、サイズが固定では
 不便なので、消去サイズを HL レジスタで
 H=横、L=縦のように指定できるようにし
 てあります。このプログラムは、List 2-3 と
 マージしてからアセンブルしてください。

なお今回のゲームでは、画面のサイズを

ゲーム座標で(0, 0)から(49, 49)までとす
 ることにしましたので、次に移動する座標
 がその範囲を越える場合には、ゼロフラグ
 を立ててからリターンするようになってい
 ます。パターンを表示する時の座標は、パ
 ターン左上の座標で示されますから、右端
 と下端はパターン・サイズを考慮に入れな
 ければなりません。そのため、右端と下端
 の値は最初からパターン・サイズの分だけ
 少なくしてあります。また、方向別の不要
 部分消去後に、移動後の座標が画面から出
 ないように、それぞれはみ出した値との比
 較を行なっています。したがって、ここで
 ゼロフラグが立てば、その座標は画面外で
 あるということになるわけです。長いプロ
 グラムといっても、内容的には同じような
 処理が 8 方向分あるだけですから、それほ
 どむずかしくはないと思います。しかし、
 ここで初めて論理演算(XOR A)が出てき
 たので説明を加えておきましょう。

論理演算については、本当に論理演算を
 するのが目的で使われる場合と、別の目的
 のために使われる場合とがあります。本格
 的な使用の説明は、適切な例が出てきた時
 にすることにして、ここではよく使用され
 るものの意味を、とりあえず覚えてくださ
 い。

XOR A: アキュムレータの値をゼロにする。フラグは変化するが、1バイトで済むために、LD A, 0 (2バイト必要)の代わりによく用いられる。

OR A: アキュムレータの値がゼロか否かを調べる。ゼロフラグの変化が、CP 0 (2バイト必要)と同じなので、その代わりに用いられる。また、キャリーフラグをリセットしたい時にも使われる。AND A も同じ意味で使われる。

さて、ここでテスト・プログラムを実行して方向別の消去がうまくされているかどうかを確認してみましょう。まず、LINE (0, 0)-(639, 20), 7, BF とでもして画面に色をつけておきます。その後で D000H 番地からテストを実行し、図 8 と同じように消去されていれば OK です。

次に、移動後の座標が正しく計算されているかどうかの確認です。それには、マシン語の実行を D00EH 番地でストップさせて、その時の BC レジスタの値を X コマンドでチェックすればいいのです。

モニタから、

```
h]GD000, D00E
h]X
```

を実行してください。B: の所が 0001 と表示されていれば B=00H, C=01H のことから、正しく次座標が計算されていることになります。同じやり方で D005H 番地の値を 01 から 08 まで順に変更し、全方向について確認をしてみてください。画面枠からハズレる場合に、表示不可能な座標とゼロフラグがセットされていることが確認できれば、このプログラムは正常に作動しているということです。

List 2-4 パターンの部分消去

```

;***** List 2-4-G *****
;
CLPTXY: ;Clear Pattern (X,Y) — (C,B)よりHLのサイズで消去
;
LD (SIZE),HL
CALL XYADR
XOR A
OUT (5CH),A
CALL ERBOX
OUT (5DH),A
CALL ERBOX
OUT (5EH),A
CALL ERBOX
OUT (5FH),A
RET

;
ERBOX: ;ERase BOX — 指定されたサイズの消去
LD HL,(DISPAD)
LD DE,HLEN
LD BC,(SIZE)

```

BE39
BE39 2269BE
BE3C CD00C0
BE3F AF
BE40 D35C
BE42 CD52BE
BE45 D35D
BE47 CD52BE
BE4A D35E
BE4C CD52BE
BE4F D35F
BE51 C9

BE52
BE52 2A37BE
BE55 115000
BE58 ED4B69BE

DISPAD

DISPAD)HLに入れるため
画面、レンド面、消去
指定されたサイズの消去
消去アドレス
消去の横方向のバイト数

```

BE5C      ERL1:  ;ERase Loop 1
BE5C C5    PUSH BC
BE5D E5    PUSH HL
BE5E      ERL2:  ;ERase Loop 2
BE5E 77    LD     (HL),A
BE5F 23    INC    HL
BE60 10FC  DJNZ   ERL2
BE62 E1    POP    HL
BE63 19    ADD    HL,DE
BE64 C1    POP    BC
BE65 0D    DEC    C
BE66 20F4  JR     NZ,ERL1
BE68 C9    RET

```

```

BE69      ;
BE69      SIZE:  ;SIZE
           DS     2

```

```

;***** List 2-4-N *****
;

```

```

C02B      MVCLS: ;MoVe CLS
C02B C5    PUSH BC
C02C 3D    DEC    A
C02D 282B  JR     Z,D1CLS
C02F 3D    DEC    A
C030 2834  JR     Z,D2CLS
C032 3D    DEC    A
C033 284C  JR     Z,D3CLS
C035 3D    DEC    A
C036 2858  JR     Z,D4CLS
C038 3D    DEC    A
C039 2873  JR     Z,D5CLS
C03B 3D    DEC    A
C03C 287F  JR     Z,D6CLS
C03E 3D    DEC    A
C03F CAD9C0 JP     Z,D7CLS

```

```

C042      ;
C042      D8CLS: ;Direction 8 CLS
C042 210404 LD     HL,404H
C045 CD39BE CALL  CLPTXY
C048 C1    POP    BC
C049 04    INC    B
C04A C5    PUSH BC
C04B 210C01 LD     HL,10CH
C04E CD39BE CALL  CLPTXY
C051 C1    POP    BC
C052 0C    INC    C
C053 CDE5C0 CALL  RCHECK
C056 C4F1C0 CALL  NZ,DCHECK
C059 C9    RET

```

```

C05A      ;
C05A      D1CLS: ;Direction 1 CLS
           LD     HL,110H

```

C 値を 0 に
0 5

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入

移動方向 1 に入


```

C05D CD39BE      CALL CLPTXY
C060 C1          POP BC
C061 0C          INC C
C062 CDE5C0      CALL RCHECK
C065 C9          RET

```

移動方向 2 の不要部分消去
 消去のサイズ
 B)よりHLのサイズで消去するため
 HLの値をスタックから取り出す
 BCの値をスタックへ退避

```

;
D2CLS: ;Direction 2 CLS
C066          LD HL,10CH
C066 210C01      CALL CLPTXY
C069 CD39BE      POP BC
C06C C1          PUSH BC
C06D C5          INC B
C06E 04          INC B
C06F 04          INC B
C070 04          LD HL,404H
C071 210404      CALL CLPTXY
C074 CD39BE      POP BC
C077 C1          DEC B
C078 05          INC C
C079 0C          CALL RCHECK
C07A CDE5C0      CALL NZ,UCHECK
C07D C4EDC0      RET
C080 C9

```

移動方向 2 の不要部分消去
 消去のサイズ
 B)よりHLのサイズで消去するため
 HLの値をスタックから取り出す
 BCの値をスタックへ退避
 移動方向 3 の不要部分消去
 消去のサイズ
 B)よりHLのサイズで消去するため
 HLの値をスタックから取り出す
 BCの値をスタックへ退避
 移動後の座標の右端チェック。右端
 と右端チェックするため

```

;
D3CLS: ;Direction 3 CLS
C081          INC B
C081 04          INC B
C082 04          INC B
C083 04          LD HL,404H
C084 210404      CALL CLPTXY
C087 CD39BE      POP BC
C08A C1          DEC B
C08B 05          INC C
C08C CDEDC0      CALL UCHECK
C08F C9          RET

```


移動方向 3 の不要部分消去
 消去のサイズ
 B)よりHLのサイズで消去するため
 HLの値をスタックから取り出す
 BCの値をスタックへ退避
 移動後の座標の右端チェック。右端
 と右端チェックのため


```


;
D4CLS: ;Direction 4 CLS
C090          INC B
C090 04          INC B
C091 04          INC B
C092 04          LD HL,404H
C093 210404      CALL CLPTXY
C096 CD39BE      POP BC
C099 C1          PUSH BC
C09A C5          INC C
C09B 0C

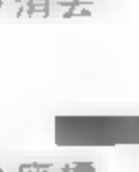
```


移動方向 4 の不要部分消去
 消去のサイズ
 B)よりHLのサイズで消去するため
 HLの値をスタックから取り出す
 BCの値をスタックへ退避



● 消去

 ● 座標
 (C,B)→(C+1,B+1)
 ■ 右端チェック
 ■ 下端チェック

● 消去

 ■ 座標
 (C,B)→(C+1,B)
 ● 右端チェック

● 消去

 ● 座標
 (C,B)→(C+1,B-1)
 ● 右端チェック
 ● 上端チェック

● 消去

 ● 座標
 (C,B)→(C,B-1)
 ■ 上端チェック

● 消去

 ■ 座標
 (C,B)→(C-1,B-1)
 ■ 左端チェック
 ● 上端チェック

21780	C09C 0C	INC C		
	C09D 0C	INC C		
	C09E 210C01	LD HL,10CH		
	C0A1 CD39BE	CALL CLPTXY		消去のサイズを消去する
	C0A4 C1	POP BC		消去のサイズをスタックから取り出す
	C0A5 05	DEC B		
	C0A6 0D	DEC C		
	C0A7 CDE9C0	CALL LCHECK		移動後の座標の左端チェックをするため
	C0AA C4EDC0	CALL NZ,UCHECK		左端チェックの際は上端チェックをする
	C0AD C9	RET		移動後の座標の左端チェックをコール
	C0AE		;D5CLS: ;Direction 5 CLS	移動方向 5 の不要部分消去
	C0AE 0C	INC C		
	C0AF 0C	INC C		
	C0B0 0C	INC C		
	C0B1 211001	LD HL,110H		
	C0B4 CD39BE	CALL CLPTXY		消去のサイズを消去する
	C0B7 C1	POP BC		消去のサイズをスタックから取り出す
	C0B8 0D	DEC C		
	C0B9 CDE9C0	CALL LCHECK		移動後の座標の左端チェックをするため
	C0BC C9	RET		
	C0BD		;D6CLS: ;Direction 6 CLS	移動方向 6 の不要部分消去
	C0BD 210404	LD HL,404H		HL・消去のサイズ  P.63 参照
	C0C0 CD39BE	CALL CLPTXY		B)よりHLのサイズで消去するため
	C0C3 C1	POP BC		消去のサイズをスタックから取り出す
	C0C4 C5	PUSH BC		消去のサイズをスタックへ返却
	C0C5 0C	INC C		
	C0C6 0C	INC C		
	C0C7 0C	INC C		
	C0C8 04	INC B		C+3
	C0C9 210C01	LD HL,10CH		HL・消去のサイズ
	C0CC CD39BE	CALL CLPTXY		B)よりHLのサイズで消去するため
	C0CF C1	POP BC		B)の値をスタックから取り出す
	C0D0 04	INC B		B+3+1
	C0D1 0D	DEC C		C=C-1
	C0D2 CDE9C0	CALL LCHECK		移動後の左端チェックをするため
	C0D5 C4F1C0	CALL NZ,DCHECK		左端チェックの際は下端チェックをする
	C0D8 C9	RET		移動後の座標の下端チェックをコール
	C0D9		;D7CLS: ;Direction 7 CLS	 P.63 参照
	C0D9 210404	LD HL,404H		移動方向 7 の不要部分消去
	C0DC CD39BE	CALL CLPTXY		HL・消去サイズ
	C0DF C1	POP BC		B)よりHLのサイズで消去するため
	C0E0 04	INC B		BCの値をスタックから取り出す
	C0E1 CDF1C0	CALL DCHECK		B+1
	C0E4 C9	RET		移動後の座標の下端チェックをするため
	002E	REND: EQU 46 ;Right END		右端座標=46
	0000	LEND: EQU 0 ;Left END		左端座標=0
22280	0000	UEND: EQU 0 ;Up END		上端座標=0

002E	DEND: EQU 46 ;Down END	——下端座標=46
	;	
C0E5	RCHECK: ;Right CHECK	——右端のチェック
C0E5 3E2F	LD A,REND+1	REND+1の時、画面右へ出過ぎ
C0E7 B9	CP C	
C0E8 C9	RET	
C0E9	LCHECK: ;Left CHECK	——左端のチェック
C0E9 3EFF	LD A,LEND-1	LEND-1の時、画面左へ出過ぎ
C0EB B9	CP C	
C0EC C9	RET	
C0ED	UCHECK: ;Up CHECK	——上端のチェック
C0ED 3EFF	LD A,UEND-1	UEND-1の時画面下へ出過ぎ
C0EF B8	CP B	
C0F0 C9	RET	
C0F1	DCHECK: ;Down CHECK	——下端チェック
C0F1 3E2F	LD A,DEND+1	DEND+1の時、画面下へ出過ぎ
C0F3 B8	CP B	
C0F4 C9	RET	
	;	
	***** List 2-4-T *****	
	;	
D000	TEST: ;TEST	——メインルーチン
D000 F3	DI	割り込み禁止
D001 3100B6	LD SP,STACK	スタックポインタを B600H に設定
D004 3E01	LD A,1	移動方向
D006 010000	LD BC,0000	現在位置(C,B)=(0,0)
D009	TLOOP: ;Test Loop	
D009 F5	PUSH AF	AF の値をスタックへ退避
D00A C5	PUSH BC	HI の値をスタックへ退避
D00B CD2BC0	CALL MVCLS	移動方向別の消去を行なったため
D00E C1	POP BC	消去前の座標をスタックから取り出す
D00F 0C	INC C	
D010 0C	INC C	
D011 0C	INC C	
D012 0C	INC C	
D013 0C	INC C	
D014 F1	POP AF	AF の値をスタックから取り出す
D015 3C	INC A	A=A+1 次の移動方向にする
D016 FE09	CP 9	
D018 20EF	JR NZ,TLOOP	OK なければ TLOOPへ
D01A FB	EI	割り込み許可
50220 D01B FF	RST 38H	モニター戻る

●消去
■座標
(C,B)⇒(C-1,B)
●左端チェック

●消去
■座標
(C,B)⇒(C-1,B+1)
■左端チェック
■下端チェック

●消去
■座標
(C,B)⇒(C,B+1)
■下端チェック

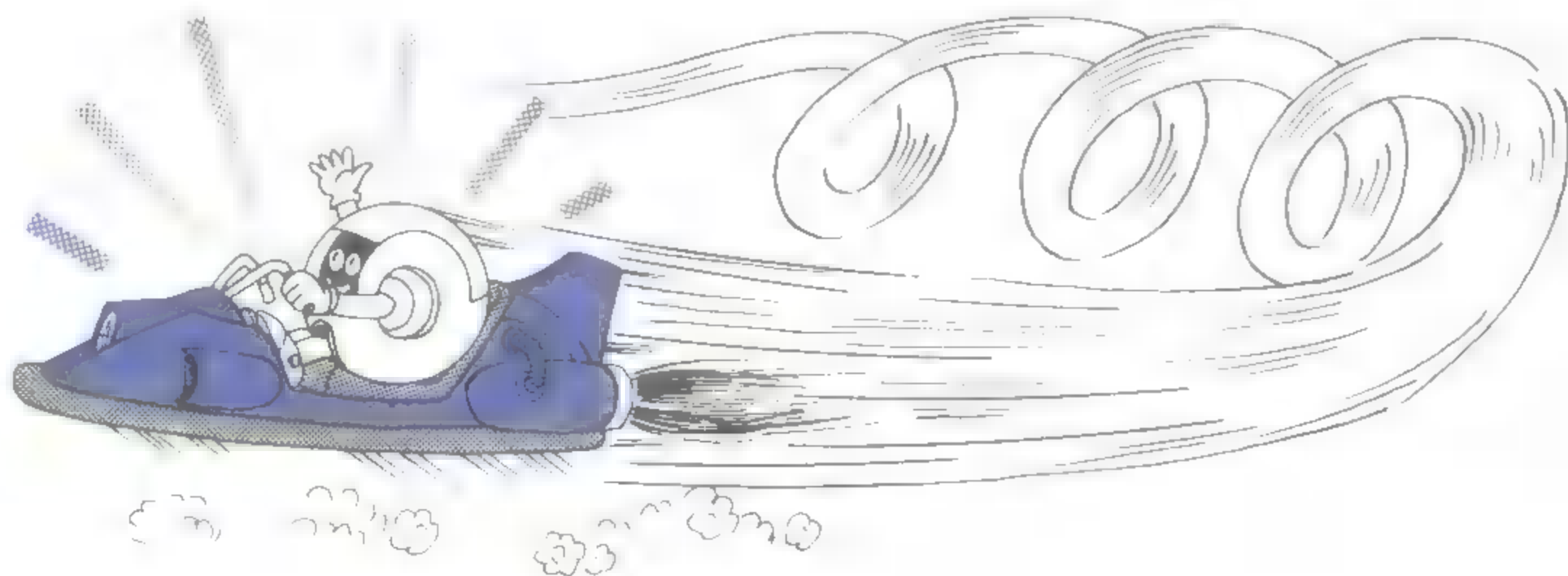
6. パターン移動…データにそって移動

パターンを動かすために必要な準備は整いました。さて、どのように動かしたらいいでしょうか。つまり、勝手に動けといってもコンピュータは命令がなければ何もできないのは、御存知の通りです。日本人の習慣で「適当にたのむ…」という言葉が、寿司屋とか酒場などでよく聞かれますが、これが通用するのは店の方で最初から「適当に」というメニューを用意しているからですね。よく考えると、こんな恐ろしい言葉はナイのですが、日本人は謙虚な人種ですから、決して一番高い料理を出して大儲けをしようなどとは誰も思わないのです。それよりも、また来てもらった方がいいということを知っているのです。

コンピュータにも、この「適当に」が通用するようになると、本当に便利なのですが、残念ながら無理なのです。そこで、まずパターンに画面の中をグルグル回ってもらうことにしました。List 2-5 を見てください。ここでの処理はすべて 50000 行からの

テスト・プログラムの中で行なわれています。基本的な考え方は BASIC でデータ文を読むのと同じことで、スタート地点から次に移動する方向をすべてデータとして用意しているのです。たったこれだけのことで、このパターンは画面の中をいつまでもグルグル回り続けることになります。これでは、終わりがなく暴走しているようなものですから、とりあえず 15 回転したならばストップするようにカウンタをつけました。

では、List 2-5 を打ち込み、List 2-3 と List 2-4 のマージされたプログラムに List 2-5 をマージしてください。なお、以降 3 章の終わりまで、同様に一つずつプログラムをマージしていくことになりますが、いちいちマージするようには書いてありませんので、かならず、打ち込んだらアスキーセーブをして、前のプログラムとマージしてからアSEMBルしてください。



このようなデータのことをテーブルともいい、うまく利用すると計算式では求められない複雑な動きを、簡単な上、高速に実現させることができます。これはキャラクタ・パターンへ性格をつける上において、ゲームでは大変有効なテクニックの1つです。なお、プログラム中のデータ作成は、このように実際の数値だけでなくラベルで代用できますから、アセンブラの使い方として覚えておくと便利です。

さて、実際にテストの実行をすると、これまでと違ってテキスト画面の邪魔な文字が消えています。それも、プログラムが終了しても消えたままですから、中には不安になった方もいるかもしれません。まず、見えるようにする方法ですが、テキスト画面は見えなくてもh]の状態で止まっていると考えて、次のように入力します。

h] ^ b... [CTRL] + B
WIDTH 80

これで、見えるようになったはずです。このテキスト画面を消すということは、単に邪魔な文字を消しているだけでなく実行速度のアップにもなっているのです。これは、テキスト画面がDMA(Direct Memory Access)によりCPUとは別に、直接メモリをアクセスして表示されるようになっているからです。すなわちDMAによりテキスト画面表示中にはCPUが止まり、その分速度の低下を招いているからです。そこで、CPUが止まらないようにDMAを止めてしまえば、テキスト画面は消えるが実行速度はアップするわけです。このDMAをオフにする方法が、出力ポート51Hに0を出力することなのです。

何だかよくわからないかもしれませんが、邪魔者が消えた上に実行速度がアップするのであれば、これはいいことに間違いありません。

List 2-5 データによるパターンの

***** List 2-5-T *****	
D000	TEST: ;TEST
D000 F3	DI
D001 3100B6	LD SP,STACK
D004 AF	XOR A
D005 0351	OUT (51H),A
D007 3E10	LD A,10H
D009 3237D0	LD (COUNT),A
D00C 011419	LD BC,1914H
D00F	TINIT: ;Test Initialize
D00F 213AD0	LD HL,DATA
D012 2238D0	LD (DATAWK),HL
D015 3A37D0	LD A,(COUNT)
D018 3D	DEC A
50150 D019 3237D0	LD (COUNT),A

HL ← DATA 5向データ・ポ
DATAWK ← HL ータの初期設定
COUNT の値を -1 する

```

D01C 2002      JR    NZ,TLOOP
D01E FB        EI
D01F FF        RST   38H

;
D020           TLOOP: ;Test LOOP
D020 2A38D0     LD    HL,(DATAWK)
D023 7E         LD    A,(HL)
D024 23         INC   HL
D025 2238D0     LD    (DATAWK),HL
D028 B7         OR    A
D029 28E4       JR    Z,TINIT
D02B CD2BC0     CALL  MVCLS
D02E C5         PUSH  BC
D02F 3E01       LD    A,1
D031 CD00BE     CALL  DISP
D034 C1         POP   BC
D035 18E9       JR    TLOOP

;
D037           COUNT: ;COUNTER
D037           DS    1
D038           DATAWK: ;DATA work area
D038           DS    2

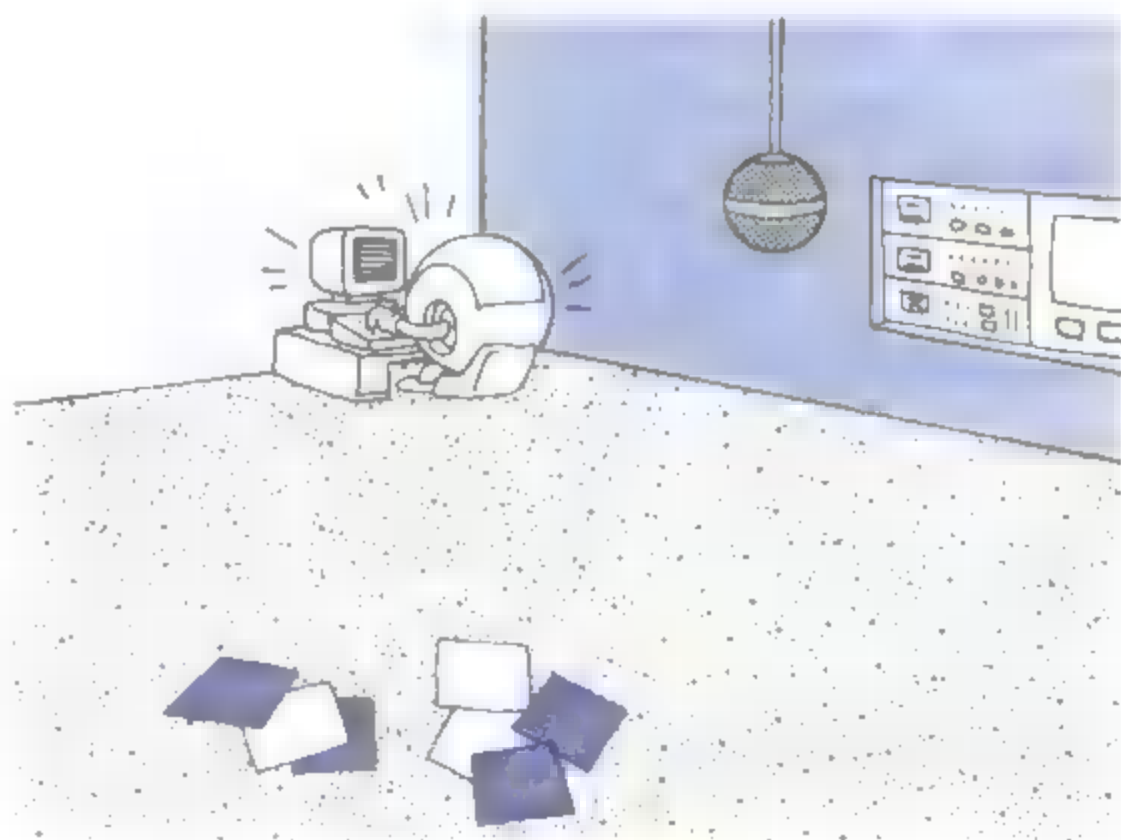
;
0001           RR:    EQU   1
0002           UR:    EQU   2
0003           UU:    EQU   3
0004           UL:    EQU   4
0005           LL:    EQU   5
0006           DL:    EQU   6
0007           DD:    EQU   7
0008           DR:    EQU   8

;
D03A           DATA: ;direction DATA
D03A 07070708   DB     DD,DD,DD,DR
D03E 07070807   DB     DD,DD,DR,DD
D042 08080801   DB     DR,DR,DR,RR
D046 08080108   DB     DR,DR,RR,DR
D04A 01010102   DB     RR,RR,RR,UR
D04E 01010201   DB     RR,RR,UR,RR
D052 02020203   DB     UR,UR,UR,UU
D056 02020302   DB     UR,UR,UU,UR
D05A 03030304   DB     UU,UU,UU,UL
D05E 03030403   DB     UU,UU,UL,UU
D062 04040405   DB     UL,UL,UL,LL
D066 04040504   DB     UL,UL,LL,UL
D06A 05050506   DB     LL,LL,LL,DL
D06E 05050605   DB     LL,LL,DL,LL
D072 06060607   DB     DL,DL,DL,DD
D076 06060706   DB     DL,DL,DD,DL
D07A 0700       DB     DD,0

```


7. 大量出現…1人じゃつままない!

1つのパターンが動けば、次は数を増やしたくなるのがこれまた人間の欲というか、心理です。諺にもありましたね…『這えば立て、立てば歩めの親心』…マア、それほど可愛さではないにしても、自分で作成したパターンが自分の思い通りに動いてくれるということは、ある種の感動があるものです。自分の子供でさえも、これほど思い通りには動いてくれませんか。それどころか、段々反抗的にさえなるのですから、画面の中のこの小さなパターンとは大違いです。もっとも、いつまでたっても命令通りのことしかできない人間では、教える方も面倒でたまりません。コンピュータも同じことで、そのために思考能力のある人工知能を開発しているくらいです。程度は違え、少しずつパターンが成長していくということは、非常にうれしいものです。



ここでのプログラムは、これまでのものに比べかなり実際のゲームを意識して作られています。つまり、この段階では不必要なものも、現実のゲーム・プログラムに近づけるために、入れてあります。そのことを頭に入れた上で、まずは大量出現のための秘密兵器、新しいレジスタの登場です。それは、インデックス・レジスタという16ビット専用のレジスタのことで、IXとIYの2種類があります。この2つは内容的にはまったくの同格で、どちらを使っても機能的な差はありません。

さて、その特徴ですが、これまでのペアレジスタと一番違う点は、自分自身の指し示すアドレスの内容を操作するだけでなく、その前後(−80Hから+7FH)のアドレスの内容をも、同じように操作できるということです。このことを具体的な例で示すと、次のようになります。

例 IXの値がD500Hで、D505H番地の中身をD50AH番地に移動するという場合

```
LD    A, (IX+5)
LD    (IX+0AH), A
```

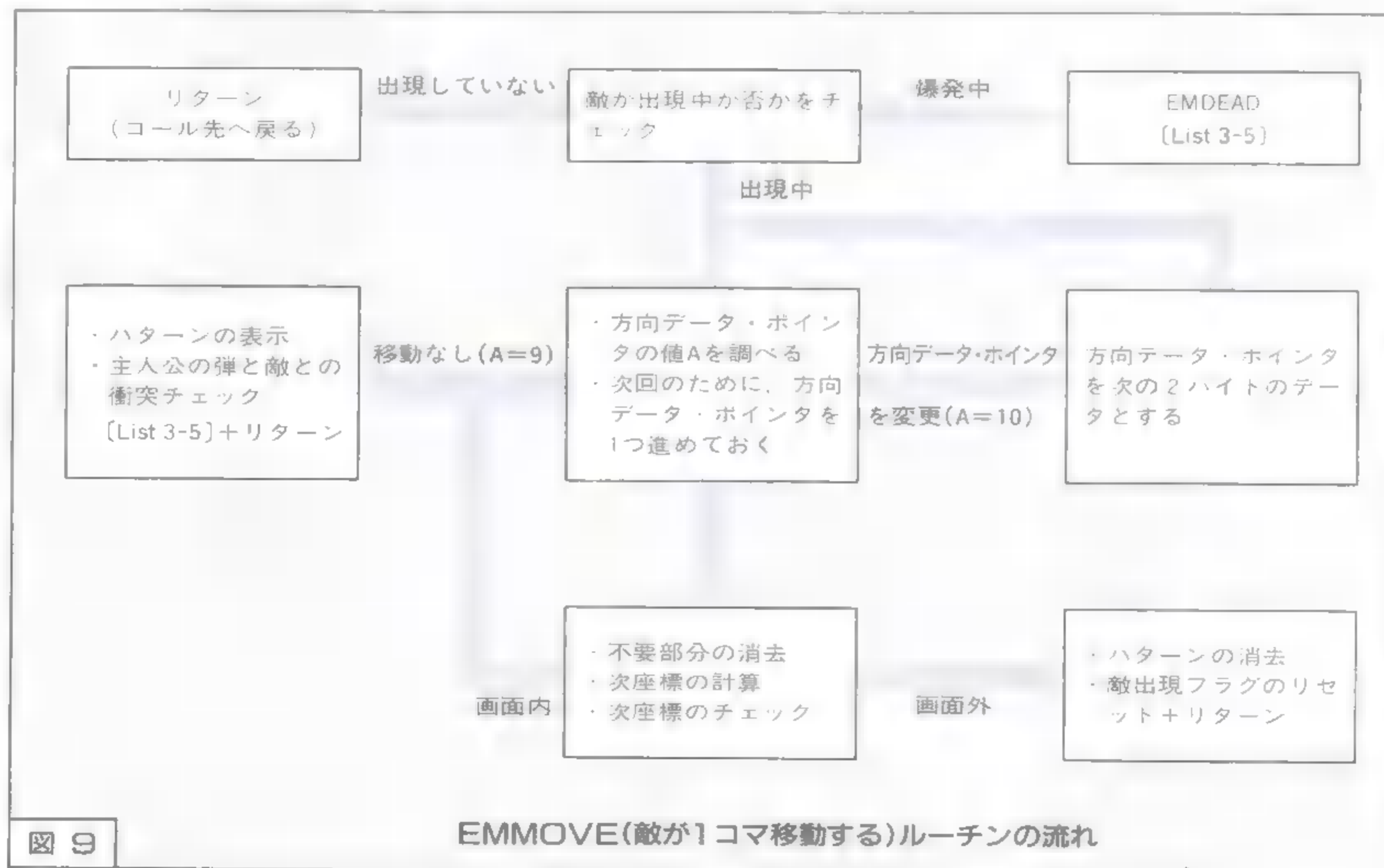
これだけでは、アドレスを絶対番地で示したのと変わりがなさそうですが、インデックス・レジスタを使用すれば、その基準となる値を変えることにより、どのアドレスでも表現することが可能ですから、その違いは天と地ほどあることになります。この特徴を利用して、ここでは3つの敵(勝

手に動くということはすなわち敵となる)を出現させ、パターンもそれぞれ変えることにしました。パターン・エディタでカラーページの④の図を参考にして、3つのデータを B6C0H, B780H, B840H 番地に作成してください。その内の1つは、すでに作ってありますから、ここで新たに作るのは2パターンということです。3つまとめたセーブ・アドレスは B6C0H~B8FFH 番地となります。

この List 2-6 に、コメント文として書かれているニーモニックがあります。これが後で(3.5章参照)使われる部分なのですが、敵のワークエリアの内容がわかると、この部分の意味も簡単に想像できると思います。そこで、まずそのワークエリアの内容を確認してみましょう。

- (IX+0): 00H…画面に出現していない
01H…画面に出現中
FFH…弾に当たって爆発中
- (IX+1): パターン番号
- (IX+2): X座標
- (IX+3): Y座標
- (IX+4): 利用している方向データ・ポインタ(下位)
- (IX+5): 利用している方向データ・ポインタ(上位)
- (IX+6): 加算される得点(下位)
- (IX+7): 加算される得点(上位)

これだけあるワークエリアの内、ここで必要なのはパターン番号、座標、方向データ・ポインタの3つだけなのですが、実際のゲーム(といっても、2,3章だけで作るミニミニ・ゲームの話)で必要になると思われるものも、用意されています。ワークエリ



アの内容とラベルの意味がわかると、プログラムというものは大変理解しやすくなるもので、コメント文のニーモニックの内容も、すでに大体の想像はついているかもしれません。ここではその部分に関して、それ以上の追求は避けます。

なお、50210～50300行の方向を示すデータの中に新たに9と10という番号が加わっています。これもここで必ずしも必要というものではないのですが、9は移動ナシ、10は方向データ・ポインタを新しく変更することを意味しています。実際には、10はこれまでの0と同じような意味ですから特に目新しいことはありません。また9はこのテストでは使用されていません。

敵が1コマ移動するまでの全体の流れは、図9のようになっています。プログラムだけを追いかけると、どうしても視野が

狭くなってしまい、全体がボケてしまうことがあります。経済学にもマクロとミクロがあるように、プログラムもマクロ(全体の内容)を考えながら、ミクロ(1行ごとの命令)を組み立てるようにしないといけないということです。

もう1つの新しいテクニックは、プログラムをストップキーによって終了させていることです。このキー・スキャンの方法については、次節で詳しく取り上げてますので、ここは早速テスト・プログラムの実行に移りましょう。3機編隊の敵が、画面の中をグルグル回り出しています。移動方向データを変えれば、色々な動きをさせることも可能ですから、遊んでみるのもいいかもしれません。しかし、まだ画面からはみ出した時の処理はされていないので、あまり無理はさせないようにお願いします。

P.S. マージとアセンブルを忘れずに。

List 2-6 パターンの大量出現

23000	***** List 2-6-N *****	
	EMMOVE: ;Enemy MOVE	——すべての敵の移動 → 参照
C0F5	LD A,(IX+0)	A ← (IX+0)
C0F5 DD7E00	OR A	A ← A
C0F8 B7	RET Z	A=0ならリターン
C0F9 C8	INC A	A ← A+1
	JP Z,EMDEAD	A=FFH 即ち A=FFHならEMDEADへ
	ORG 0C0FEH	書いておくとアドレスが変わらない
	LD C,(IX+2)	C ← (IX+2) ← X座標
23120 C101 DD4603	LD B,(IX+3)	B ← (IX+3) ← Y座標

敵出現フラグの内容
 (IX+0)=00H → 出現していない
 (IX+0)=01H → 出現中
 (IX+0)=FFH → 爆発中

```

C104 DD6E04      LD      L,(IX+4)
C107 DD6605      LD      H,(IX+5)
C10A              EM0: ;Enemy Move-0
C10A 7E          LD      A,(HL)
C10B 23          INC     HL
C10C DD7504      LD      (IX+4),L
C10F DD7405      LD      (IX+5),H
C112 FE09        CP      9
C114 280F        JR      Z,EM1
C116 FE0A        CP      10
C118 282A        JR      Z,EM2
C11A CD2BC0      CALL    MVCLS
C11D 282B        JR      Z,EMFIN
C11F DD7102      LD      (IX+2),C
C122 DD7003      LD      (IX+3),B
C125              EM1: ;Enemy Move
C125 DD7E01      LD      A,(IX+1)
C128 CD00BE      CALL    DISP
;                CALL    EMCHK
;                RET      NC
;                DEC     HL
;                LD      (HL),0
;                LD      (IX+0),0FFH
;                LD      (IX+1),EXPLO1
;                LD      E,(IX+6)
;                LD      D,(IX+7)
;                CALL    DISPSC
;
;                ORG     0C143H
;
C143 C9          RET
C144              EM2: ;Enemy Move 2
C144 5E          LD      E,(HL)
C145 23          INC     HL
C146 56          LD      D,(HL)
C147 EB          EX      DE,HL
C148 18C0        JR      EM0
;
C14A              EMFIN: ;Enemy Move FINish
C14A DD4E02      LD      C,(IX+2)
C14D DD4603      LD      B,(IX+3)
C150 211004      LD      HL,410H
C153 CD39BE      CALL    CLPTXY
C156 DD360000    LD      (IX+0),0
C15A C9          RET
;
0003              EMVAL: EQU 3 ;Enemy VALue——敵の回数
;
C15B              EMMVAL: ;Enemy Move ALI
C15B DD216EC1    LD      IX,EMWORK
C15F 0603        LD      B,EMVAL

```



```

C161      EMALP: ;Enemy Move All Loop
C161 C5    PUSH BC
C162 CDF5C0 CALL EMMOVE
C165 111000 LD DE,16
C168 DD19  ADD IX,DE
C16A C1     POP BC
C16B 10F4   DJNZ EMALP
C16D C9     RET

;
C16E      EMWORK: ;Enemy WORK area
C16E      DS 48

;
;***** List 2-6-T *****
;
D000      TEST: ;TEST
D000 F3     DI
D001 3100B6 LD SP,STACK
D004 AF     XOR A
D005 0351   OUT (51H),A

;
D007 215FD0 LD HL,INITDT
D00A 116EC1 LD DE,EMWORK
D00D 013000 LD BC,48
D010 EDB0   LDIR

;
D012      TLOOP: ;Test LOOP
D012 CD58C1 CALL EMMVAL
D015 DB09   IN A,(9)
D017 1F     RRA
D018 38F8   JR C,TLOOP
D01A FB     EI
D01B FF     RST 38H

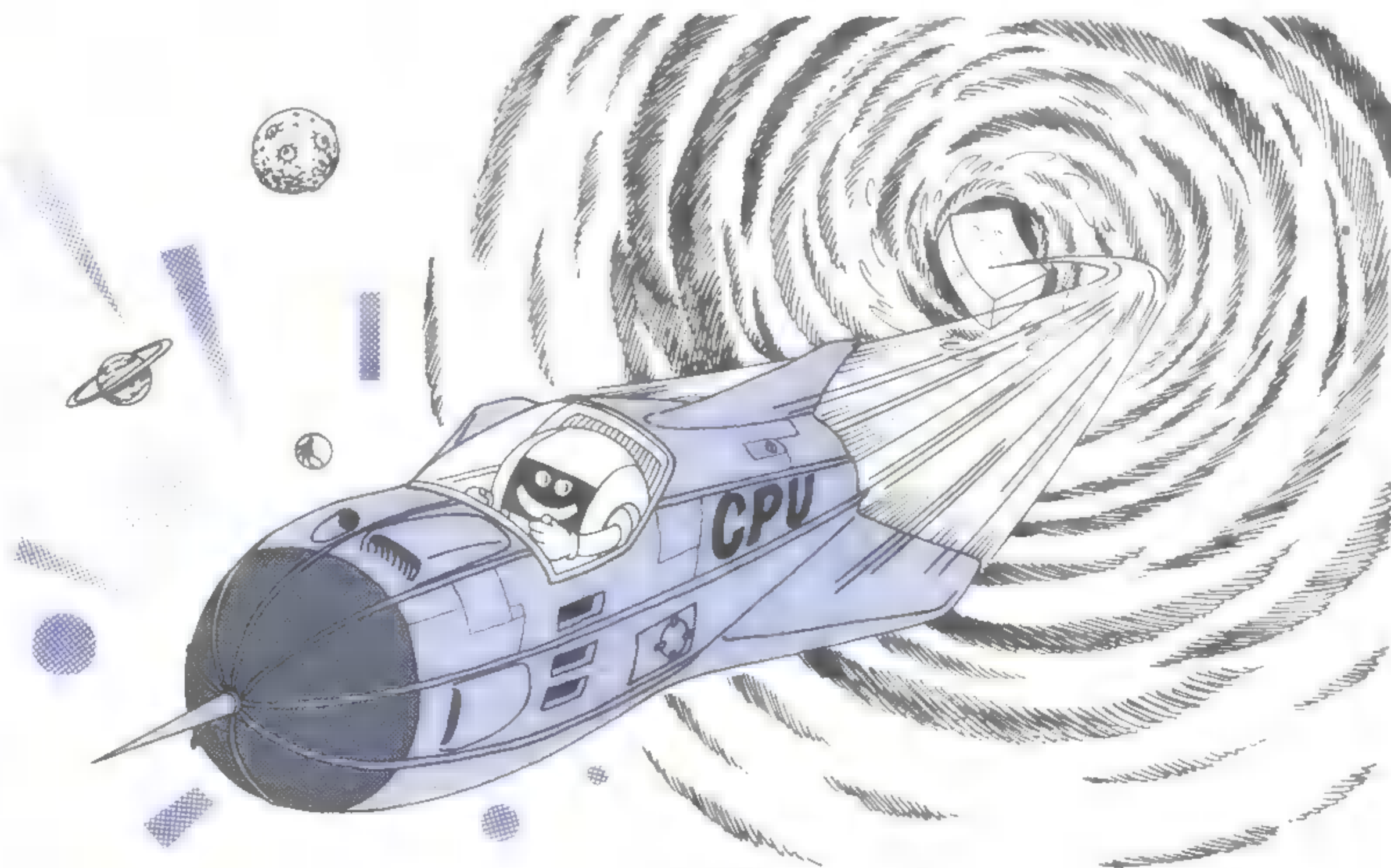
;
0001      RR: EQU 1
0002      UR: EQU 2
0003      UU: EQU 3
0004      UL: EQU 4
0005      LL: EQU 5
0006      DL: EQU 6
0007      DD: EQU 7
0008      DR: EQU 8
0009      NM: EQU 9 ;No Movement
000A      NP: EQU 10 ;New Pointer

;
D01C      DATA: ;direction DATA
D01C 07070708 DB DD,DD,DD,DR
D020 07070807 DB DD,DD,DR,DD
D024 08080801 DB DR,DR,DR,RR
D028 08080108 DB DR,DR,RR,DR
D02C 01010102 DB RR,RR,RR,UR
D030 01010201 DB RR,RR,UR,RR

```

——移動しない
——次の2バイトを方向データポインタとする
——移動方向を示すデータ

50390	D034 02020203	DB	UR,UR,UR,UU	
	D038 02020302	DB	UR,UR,UU,UR	
	D03C 03030304	DB	UU,UU,UU,UL	
	D040 03030403	DB	UU,UU,UL,UU	
	D044 04040405	DB	UL,UL,UL,LL	
	D048 04040504	DB	UL,UL,LL,UL	
	D04C 05050506	DB	LL,LL,LL,DL	
	D050 05050605	DB	LL,LL,DL,LL	
	D054 06060607	DB	DL,DL,DL,DD	
	D058 06060706	DB	DL,DL,DD,DL	
	D05C 0A	DB	NP	
	D05D 1CD0	DW	DATA	
	D05F			
	D05F 01010B10	DB	1,1,0BH,10H	
	D063 1CD0	DW	DATA	
	D065	DS	10	
	D06F 0102011F	DB	1,2,1,1FH	
	D073 1CD0	DW	DATA	
	D075	DS	10	
	D07F 0103151F	DB	1,3,15H,1FH	
	D083 1CD0	DW	DATA	
50610	D085	DS	10	



8. キー入力…コントロール&ショット

画面の中を勝手に動いているパターンは、どうやっても主人公にはできませんから、必然的に敵ということになります。しかし、時には主人公が勝手に動いて、こちらはその他大勢いる敵、というゲームがあってもいいと思うのですが…。大体、普通のゲームでは、敵はいくらでもいるのに、こちらは多くても5人または5機というふうに、大変なハンディを背負っているわけです。これでは、ゆとりを持ってゲームを楽しむことはできません。ここでいう『ゆとり』とは、例えばテレビの実況生放送では、放送終了間際になるといつ「放送時間がなくなりましたので、大変残念ですが…」となるか不安ですが、録画放送ならば安心して楽しみながら見ていられる、というようなものです。わかる人にしかわからない、飛躍した例になってしまいましたが、せめて自作のゲーム位は死なないようにするとかして、『ゆとり』を持ちたいものですね。

さて、やはり敵ばかりではゲームとして成立しませんので、キー操作によってコントロールできる主人公が必要です。ついでに、敵を倒すための小道具として弾と、やられた時、あるいは敵を倒した時の爆発パターンも作成しておくことにしましょう。パターンはカラーページの④の通りです。作成したデータは、指定のアドレスに転送し、敵のパターン・データと合わせ、まとめて1つのグラフィック・データとしてセーブします。なお、弾のサイズは横8ドット

(1バイト)なので、パターン・データは他のものと違い(B・R・G, B・R・G, …)の順になっている方が、表示する際にアドレス計算が少なくて済みます。そのため、弾だけはエディット終了時に、データのタイプを2とするようにしてください。

では、List 2-7 を一通り読んでください。

キー操作によって主人公を動かすには、どのキーが押されたかを判定できればいいのですが、このように外部からの入力に対して、その受付をする窓口のことを入力ポートといいます。キーボード以外にもライトペンやデータ・レコーダーなど、すべての入力データはこの入力ポートを通してコンピュータに入ってきます。ちょうど、税関の入国管理窓口に当たるようなものです。この入力ポートの値を調べることで、どのような入力があったかを判定することができるのです。そして、キーボードからの入力データはポート番号 00H ~ 0BH で、その内容は次ページの図 10 のようになっています。

各ポートにある入力データの内容は、押されているキーのビットが0、押されていないキーのビットが1になるようになっています。入力ポートのデータは、IN 命令を使ってレジスタに取り入れることができますから、取り入れた数字をビット・チェックすれば、押されたか否かの判定ができるのです。

例 4 キーのチェック

IN A, (0) ; 入力ポート 00H の値をアキュムレータに取り入れる
 BIT 4, A ; アキュムレータのビット 4 をチェックする
 JR Z, xxxx ; ゼロフラグが立っている、すなわち押されていれば xxxx へジャンプ

入力ポート別キーボードマップ

		ビット							
		7	6	5	4	3	2	1	0
入 力 ポ ー ト	00H	7	6	5	4	3	2	1	0
	01H	RETURN	.	,	=	+	*	9	8
	02H	G キ	F ハ	E イ	D シ	C ソ	B コ	A チ	「 .
	03H	O ラ	N ミ	M モ	L リ	K ノ	J マ	I ニ	H ク
	04H	W テ	V ヒ	U ナ	T カ	S ト	R ス	Q タ	P セ
	05H	--= ホ	^ ヘ] J ム	¥ ー	[ー	Z ツ ツ	Y ン	X サ
	06H	7 ヤ	6 オ	5 エ	4 ウ	3 ア	2 フ	1 マ	0 ワ
	07H	ー ロ	？ メ	＞ ル	＜ ネ	＋ レ	■ ケ) ヨ ヨ	8 ユ ユ
	08H	CTRL	SHIFT	カナ	GRAPH	INS DEL	→	↑	HOME CLR
	09H	ESC	SPACE	f・5	f・4	f・3	f・2	f・1	STOP
	0AH	CAPS LOCK			COPY	HELP	←	↓	HTAB
	0BH							ROLL DOWN	ROLL UP

図 10

このような方法で、キー・スキャンをしていくのです。最上位ビットや最下位ビットのチェックにはストップキーの判定でやったように、ビット・シフトをしてその値をキャリーフラグに入れて判定をします。こうすれば判定が1バイトで済むことになり、わずかな節約ですがマシン語の場合よく使われます。また、ここでは弾の連続撃ちができないよう、前のキー・データを保存しておいて、押し直しがあった時だけ有効とするなど、実戦的なテクニックも入れています。スペースバーを叩き過ぎてキーボードが壊れては困りますから、スペースバーとシフトキーが同じ役目を果たすようになっています。スペースバーとシフトキーはポートは違いますがどちらもビット6によって判定されます。この判定は、24390～24480 行のスペース、シフトキー・チェック(SSKCK)で行なっています。そこで、両ポートのANDを取ってからビット6のチェックをしているのですが、理解を深めるためにここで論理演算について、その効果を簡単にマスターしましょう。

論理演算には、OR、AND、XORの3種類があり、それぞれアキュムレータの値とレジスタの値または1バイトの数字を2進数に変換した上で、ビットごとの演算を行なうものです。できた数字はアキュムレータに入ります。

- (1) OR(論理和)：どちらかのビットが1ならば、演算結果のビットを1にする。両方のビットが0ならばそのビットを0にする。一般に、アキュムレータのある特定のビットを1にしたい時に用いられること

が多い。

例：アキュムレータ=6BHとBレジスタ=C2HのORをとる

```

        6BH = 01101011
OR   C2H = 11000010
-----
        11101011 = EBH
        アキュムレータの値となる

```

- (2) AND(論理積)：両方のビットが1ならば、演算結果のビットを1にする。どちらかのビットが0ならばそのビットを0にする。
ORとは逆に、アキュムレータの特定のビットをかならず0にしたい時に用いられることが多い。

例：アキュムレータ=D2Hと1バイトの数字=07HのANDをとる

```

        D2H = 11010010
AND   D7H = 00000111
-----
        00000010 = 2H
        アキュムレータの値となる

```

- (3) XOR(排他的論理和)：両方のビットが同じなら演算結果のビットを0、違っていたらそのビットを1にする。主に、アキュムレータのある特定のビットを反転(1なら0に、0なら1に)させる時に使われる。XORは2度繰り返すと、元の値に戻るという特徴がある

例：アキュムレータ=2FHとCレジスタ=16HのXORをとる

```

        2FH = 00101111
XOR   16H = 00010110
-----
        00111001 = 39H
        アキュムレータの値となる

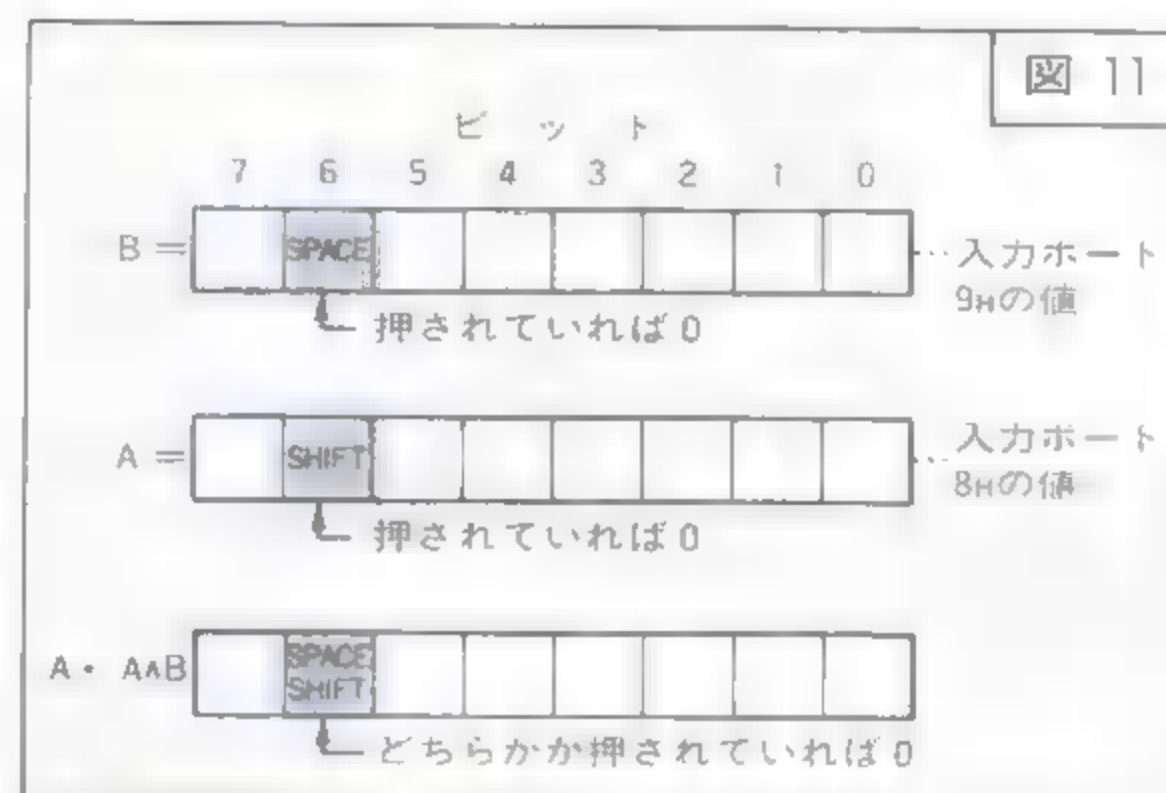
```

この他に、NOT(アキュムレータの全

ビットを反転させる演算)として、CPL という命令がありますが、めったに使うことはないと思います。また、論理演算はこのように実際に演算した結果が欲しい場合と、ビット・チェックの代用として単にフラグの変化を見るためにする場合とがあります。もちろん、両方を目的として使用することもできますから、利用次第では大変便利なものといえます。ですから、プログラム中に論理演算がでてきた場合には、まず何のために論理演算をしているのか、その使用目的を把握することが、プログラムを理解する上で大切なポイントになるのです。

さて、本プログラムのSSKCK で使われている AND ですが、スペースバーまたはシフトキーが押されていれば、どちらかの入力ポートのビット 6 は 0 になっています。そこで、両入力ポートの AND をとることにより、どちらが押されてもアキュムレータのビット 6 は 0 となります。その後で、アキュムレータのビット 6 をチェックすれば、スペースバーとシフトキーのチェックを一度にすることができるのです(図 11 参照)。

なお、ここでは弾は一度に 2 発、トータルで画面上には 6 発まで出るように設定しています。また、連続撃ちはあえてできないようになっています。プログラムの観点からは、このような制限をつけない方が簡単なのですが、キー操作に関するテクニックの 1 つとして理解するようにしてください。この判定に使われているのが、25220 行の OLDKEY というワークエリアで、先ほどの



キー・チェックでスペースバーもシフトキーも押されていなかった場合にだけ、FFHが入るようになっていました。そして、この OLDKEY の値が FFH でない場合は、たとえキーが押されていても、弾の発射はできないようにしているのです。発射 OK となった場合には、入力キーの値(ビット 6 は 0、すなわち FFH ではない)を入れて、連続撃ちを防止しているわけです。

こうして、キーにおける弾の発射が OK となっても、次に弾の総数(6 発)による制限が待っています。つまり、弾のワークエリアに空きがあるかどうかですが、これを調べているのが 24570 行の BAPOS の部分です。これは、弾のワークエリアの内容がわかれば簡単だと思いますが、弾は一度に 2 発出ることになっているため、この空きエリア捜しも 2 度実行されなければなりません。そこで、C レジスタに弾発射位置のオフセット値(X 座標用)を設定してから、BAPOS をコールしているのです。

なぜ、単純に 2 発まとめて発射していないかというと、将来敵に弾が当たった場合に 2 発とも敵に命中するとは限らず、1 発だけ当たった時は、その当たった 1 発のワークエリアだけが空きとなるからです。

なお、弾のワークエリアの内容は、次のようになっています。

- (HL) : 弾の出現フラグ
 ... 1 = 出現中,
 0 = 出現していない
- (HL+1) : 弾の X 座標
- (HL+2) : 弾の Y 座標

論理演算とキーボードからの入力方法がわかれば、2章はもう卒業です。テストを実行して、しばらくは楽しんでください。弾を打つには **SHIFT** か **SPACE**，右へ移動するのは **6**，左へは **4** です。エッ！もう飽きたって？ それでは先に進んでください。

List 2-7 キー入力による移動と弾の発射

12000	***** List 2-7-G *****	
	;	
BE6B	BLPUT: ;Bullet PUT	---弾の表示
BE6B CD00C0	CALL XYADR	HL: 表示アドレスを求める
BE6E 1180BA	LD DE,BDATA	DE: 弾のワークエリアのあるアドレス
BE71 ED738FBE	LD (BPSP+1),SP	スタックポインタを BPSP+1 に退避
BE75 315000	LD SP,HLEN	HLEN: 画面ラインへの増加バイト数
BE78 0604	LD B,4	B: 1 行のドット数
BE7A	BPLP: ;Bullet Put Loop	
BE7A D35C	OUT (5CH),A	---画面にバンク切り換え
BE7C 1A	LD A,(DE)	
BE7D 77	LD (HL),A	弾のデータを表示アドレスに入れ、スタックポインタを+1 する
BE7E 13	INC DE	
BE7F D35D	OUT (5DH),A	---画面にバンク切り換え
BE81 1A	LD A,(DE)	
BE82 77	LD (HL),A	
BE83 13	INC DE	
BE84 D35E	OUT (5EH),A	---画面にバンク切り換え
BE86 1A	LD A,(DE)	
BE87 77	LD (HL),A	
BE88 13	INC DE	
BE89 39	ADD HL,SP	HL: 表示アドレス
BE8A 10EE	DJNZ BPLP	---8 回、弾のドット数だけ繰り返す
BE8C D35F	OUT (5FH),A	---メイン RAM へバンク切り換え
BE8E	BPSP: ;Bullet Put Stack Pointer	
BE8E 310000	LD SP,0000	退避していたスタックポインタの値を元に戻す
BE91 C9	RET	
12000	;	
24000	***** List 2-7-N *****	
	;	
C19E	KEY: ;KEY scan	---キースキャン
C19E DB00	IN A,(0)	A: 入力ポート 0H の値
C1A0 CB67	BIT 4,A	
24050 C1A2 2806	JR Z,KEY4	A のビット 4 が 1 なら KEY4 へ

```

C1A4 CB77      BIT    6,A
C1A6 2809      JR     Z,KEY6
C1A8           NOMOVE: ;NO MOVE
C1A8 AF        XOR    A
C1A9 C9        RET
C1AA           KEY4: ;KEY 4
C1AA 3E00      LD     A,LEND
C1AC 91        SUB    C
C1AD C8        RET    Z
C1AE 3E05      LD     A,5
C1B0 C9        RET
C1B1           KEY6: ;KEY 6
C1B1 3E2E      LD     A,REND
C1B3 91        SUB    C
C1B4 C8        RET    Z
C1B5 3E01      LD     A,1
C1B7 C9        RET

;
C1B8           MYMOVE: ;MY MOVE
C1B8 ED4B36C2  LD     BC,(MYLOC)
C1BC CD9EC1    CALL   KEY
C1BF 2807      JR     Z,SKPCLS
C1C1 CD2BC0    CALL   MVCLS
C1C4 ED4336C2  LD     (MYLOC),BC
C1C8           SKPCLS: ;SKIP CLS
C1C8 AF        XOR    A
C1C9 CD00BE    CALL   DISP
C1CC C9        RET

;
002E          BULSTY:EQU 46      ;Bullet Start Y — 弾発射時のY座標
0006          BULVAL:EQU 6      ;Bullet VALUE — 弾の総数
BA80          BDATA: EQU 0BA80H ;Bullet DATA — 弾のパターン・データのあるアドレス

C1CD          SSKCK: ;Space ■ Shift Key Check
C1CD 2135C2    LD     HL,OLDKEY
C1D0 DB09      IN     A,(9)
C1D2 47        LD     B,A
C1D3 DB08      IN     A,(8)
C1D5 A0        AND    B
C1D6 CB77      BIT    6,A
C1D8 2803      JR     Z,PUSHKY
C1DA 36FF      LD     (HL),0FFH
C1DC C9        RET
C1DD           PUSHKY: ;PUSH Key
C1DD 46        LD     B,(HL)
C1DE 04        INC    B
C1DF C0        RET    NZ
C1E0 77        LD     (HL),A
C1E1 0E00      LD     C,0
C1E3 CDE8C1    CALL   BAPOS
C1E6 0E03      LD     C,3

```


C1E8	BAPOS: ;Bullet Appear POSSibility	
C1E8 2138C2	LD HL,BULWOK	弾のワ
C1E8 0606	LD B,BULVAL	弾の座標
C1ED	BALOOB: ;Bullet Appear LOOP	
C1ED 7E	LD A,(HL)	弾の座標
C1EE B7	OR A	
C1EF 2806	JR Z,BULAP	
C1F1 23	INC HL	
C1F2 23	INC HL	
C1F3 23	INC HL	
C1F4 10F7	DJNZ BALOOB	
C1F6 C9	RET	
C1F7	BULAP: ;BULlet APpear	
C1F7 3601	LD (HL),1	弾の座標
C1F9 23	INC HL	
C1FA 3A36C2	LD A,(MYLOC)	弾の座標
C1FD 81	ADD A,C	
C1FE 77	LD (HL),A	弾の座標
C1FF 23	INC HL	
C200 362E	LD (HL),BULSTY	弾の座標
C202 C9	RET	
C203	; ABMOVE: ;All Bullet MOVE	すべての弾の移動
C203 2138C2	LD HL,BULWOK	弾のワ
C206 0606	LD B,BULVAL	弾の座標
C208	BMLOOP: ;Bullet Move LOOP	
C208 7E	LD A,(HL)	弾の座標
C209 E5	PUSH HL	弾の座標をスタックへ退避
C20A B7	OR A	
C20B C415C2	CALL NZ,BMOVE	弾の座標をスタックへ退避
C20E E1	POP HL	弾の座標をスタックから取り出す
C20F 23	INC HL	
C210 23	INC HL	
C211 23	INC HL	
C212 10F4	DJNZ BMLOOP	弾の座標が1, BMLOOPを繰り返す
C214 C9	RET	
C215	; BMOVE: ;Bullet MOVE	
C215 C5	PUSH BC	弾の座標をスタックへ退避
C216 23	INC HL	HL = HL + 1
C217 4E	LD C,(HL)	弾の座標
C218 23	INC HL	HL = HL + 1
C219 46	LD B,(HL)	弾の座標
C21A E5	PUSH HL	弾の座標をスタックへ退避
C21B 210401	LD HL,104H	弾の座標
C21E CD39BE	CALL CLPTXY	弾の座標をスタックへ退避
C221 E1	POP HL	弾の座標をスタックから取り出す
C222 7E	LD A,(HL)	弾の座標
C223 B7	OR A	
C224 2809	JR Z,BLDSAP	弾の座標
C226 35	DEC (HL)	弾の座標

25090	C227 46	LD B,(HL)	B ← (HL) 弾の X 座標
	C228 2B	DEC HL	HL ← HL-1
	C229 4E	LD C,(HL)	C ← HL 弾の X 座標
	C22A CD6BBE	CALL BLPUT	弾の X 座標を表示
	C22D C1	POP BC	BC の値をスタックから取り出す
	C22E C9	RET	
		; BLDSAP: ;Bullet DiSAPpear	
	C22F	DEC HL	
	C22F 2B	DEC HL	
	C230 2B	LD (HL),0	出現フラグを 0 にする
	C231 3600	POP BC	BC の値をスタックから取り出す
	C233 C1	RET	
	C234 C9		
		; OLDKEY: ;OLD pressed KEY	
	C235	DB 0	前回押されたキーのデータ
	C235 00		
	C236	MYLOC: ;MY LOCation	主人公の座標
	C236	DS 2	
	C238	BULWOK: ;BULlet WOrK area	弾のワークエリア
	C238	DS 18	
25290		; ***** List 2-7-T *****	
50000		; TEST: ;TEST	
	D000	DI	
	D000 F3	LD SP,STACK	スタックポインタを B600H に設定
	D001 3100B6	XOR A	A を 0 にするため
	D004 AF	OUT (51H),A	51H へ A の値を出力
	D005 0351	LD HL,2E1AH	HL ← 2E1AH 主人公の初期出現座標
	D007 211A2E	LD (MYLOC),HL	主人公の座標を MYLOC に設定
	D00A 2236C2	LD HL,BULWOK	弾のワークエリアの先頭アドレス
	D00D 2138C2	LD B,BULVAL	弾の座標を B に設定
	D010 0606	TL: ;Test Loop	
	D012	LD (HL),0	弾の出現フラグを 0 にする
	D012 3600	INC HL	HL ← HL+1
	D014 23	INC HL	HL ← HL+1
	D015 23	INC HL	HL ← HL+1
	D016 23	DJNZ TL	弾の座標が 0 でない限り TL を繰り返す
	D017 10F9		
		; TMAIN: ;Test MAIN loop	
	D019	CALL MYMOVE	MYMOVE をコールし主人公を移動
	D019 CDB8C1	CALL SSKCK	SPACE + SHIFT をチェックするため
	D01C CDCDC1	CALL ABMOVE	弾の座標を移動するため
	D01F CD03C2	IN A,(9)	入力ポート 9H の値を読み込む
	D022 DB09	RRCA	右回転して、キャリービットを最上位ビットにする
	D024 0F	JR C,TMAIN	キャリービットが 1 ならば TMAIN へジャンプ
	D025 38F2	EI	中断マスクをクリア
	D027 FB	RST 38H	38H の中断を発生させる
50260	D028 FF		

コラム

最近、Apple や Atari のゲーム・ソフトが、PC8801mkIISR を代表とする made in Japan のマシンに移植されている。コンピュータ・ゲームにおける世界最大のマーケットであり消費国である U.S.A で鍛えられたソフトウェアが手近にあるマシンで動くのである。しかし、移植されたソフトを見ると必ずしも移植されたほうが良かったとは言いきれない。

いくら Apple や Atari のゲーム・ソフトが優れていると言っても、何年も前のソフトは、最新のソフトに比較すれば、やはり今一步の感はぬぐえない。もちろん時代を超えた Classic なソフトもたくさんあるけどネ。それに、中には、移植が悪いため本来のおもしろさを十分伝えていないものもある。

そこで、ゲーム・ソフトを作ろうと思うプログラマーやティッキーへ。もしチャンスがあれば、from U.S.A のソフトにふれて見てほしい。そんなわけで、私たち Suis-je と OH! KUMA は、現在オモシロイと思っているソフトを選んでみた。

■ ナイト・ミッション (Apple版)

数あるピンボール・ソフトの中で、美しさと言ひ、ボールのアクションと言ひファンタスティックである。他に PC8801 にも移植されたミッドナイト・マジックというソフトがあるが、これとは別ものである。ミッドナイト・マジックのほうは、シンプルなだけ、頭を使ったプレイが要求されるが、PC で動くアクション・ゲームの中でも

十分たのしめるソフトの1つであろう。

■ ロードランナー (Apple/PC88版)

このソフトについては、別に言うこともない。ファミリー・コンピュータ版を除き、どの機種でプレイしてもオリジナルのパズル & スリリングなおもしろさを損なっていないのはすごい。

■ ゾーク / ウィザードリィ (Apple版)

この偉大な2本のファンタジー・ゲームは、テキスト・アドベンチャーとロールプレイング・ゲームという違いはあるがどちらも、ゲームの設定がウェル・バランスで、どんな機種に移植しても楽しめるソフトとなるだろう。誰か移植してくれませんか！

P.S. 国産RPGの旗手、BPSがんばれと言いたい。

■ ボール・ブレイザー (Atari 版)

原則的には、2人でプレーするアクション・ゲーム。もちろん1人でもできるけどね。高速なエアカーで、ボールを取り合ってゴールにシュートするサッカーのようなゲームだが、画面を上下に分け、それぞれのプレーヤーから見たフィールドを3Dで表示するうえその動きの速いこと。さすがアタリ!! ただし Apple 版は…?

他にもアーコンII、ミュール、クエスト、マスカレード、Rogue と掲げればきりが無い。ゼビウスやバックマンなどのアーケード・ゲームの移植ものを除くと、国産ソフトは、MSX、ファミコンに良いものが多い。PC8801mkIISR。もっと楽しめるソフトが増えることを願い……end.

●衝突と得点計算

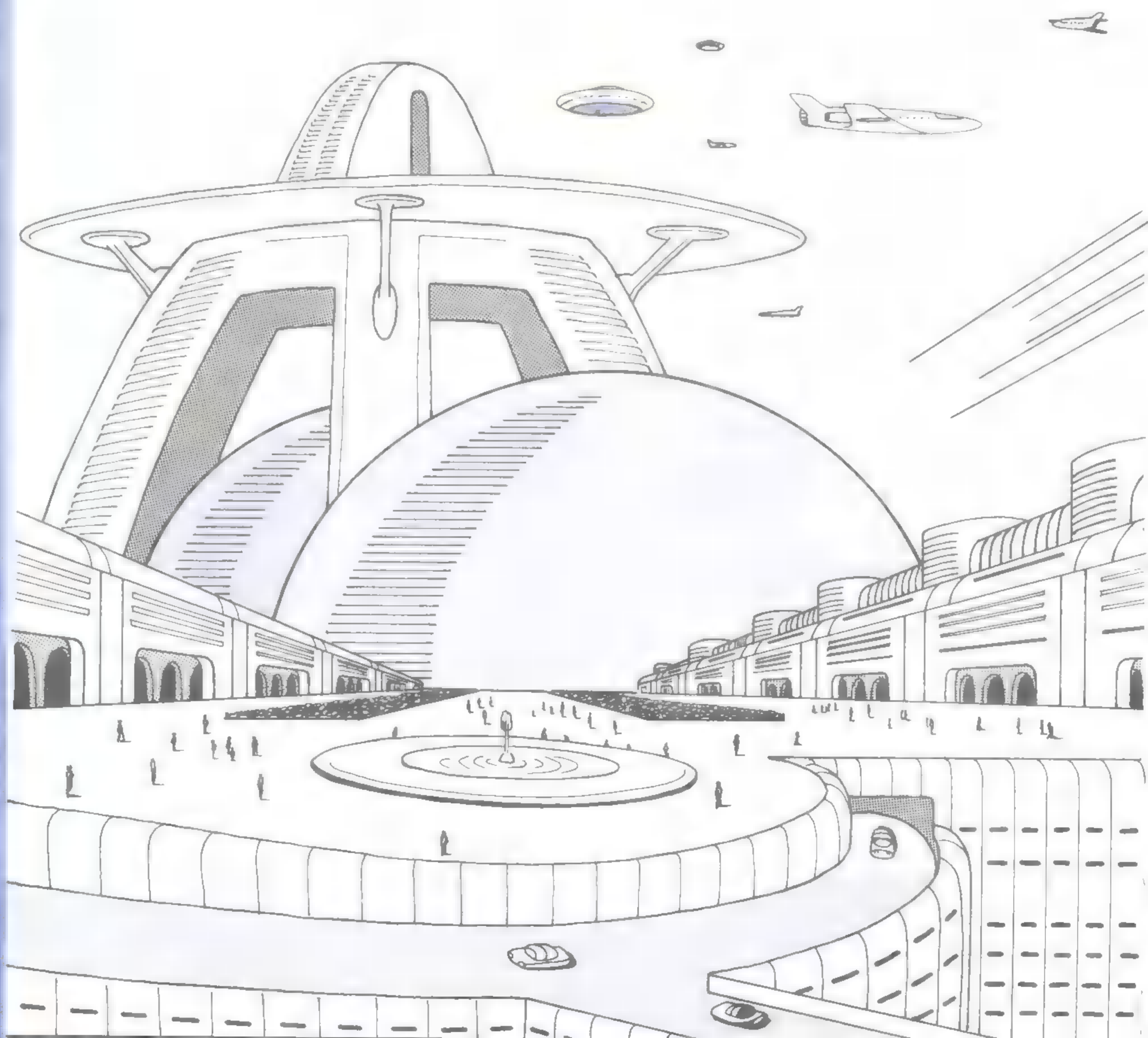
1. 衝突の判定…ゲーム座標を用いる

2. 数字…文字と数字パターンの作製

3. 計算…得点の計算と表示 その1

4. BCD…得点の計算と表示 その2

5. 衝突の処理…ゲームらしさの追求





●人間という動物は、何にでも優劣をつけたがるもので、一般社会では給料や肩書きによって差をつけていますし、学生社会においては試験による順位があります。その結果、社会での自分の位置付けなるものを自分自信でサトリ? 「まアこんなところでいいや」なんてあきらめと、中流意識が入り混じると。

●特に、コンピュータ・ゲームに中流意識やあきらめは通用しません。ハイゲームを出したい人は、スペース・バーが折れようとゲームを続け、そのゲームをキワメくではありません。それもいいな人は、自分でオリジナルなゲームを作ってしましましょう。自分の作ったゲームについては、すべて知っているワケだから…/

■と言うわけで、本章では、2章で作った表示や移動ルーチンに衝突の判定と得点計算を付け加えミニ・シューティング・ゲームを作ります。このシューティング・ゲームは、マシン語の勉強用にスーパー・サブセットとなっていますので「なアんだ、おもしろくないなアー」と言うのは、5章、6章のゲームを経験したにしてください。

1. 衝突の判定…ゲーム座標を用いる

スポーツにも色々な種類がありますが、審判の手によって勝敗がつけられるものがたくさんあります。しかし、およそ人間の判定というもののほど、あいまいで不確実なものはありません。審判が、単なる記録係の存在であるならば、そこには文句のつけようがない勝負の事実が存在するのですが、体操やボクシングの判定などのように審判員の主観が入る場合には、正確な判定を要求すること自体、最初から無理があるわけです。それで、「審判の判定は神聖である」ということにして、スポーツを成り立たせることにしたのです。この人間による審判の量たるものは、何ととっても裁判ですが、科学技術を駆使した現代でも、誤審は避けられないのが実情のようです。

2章の最後のプログラムで、動きとしてはかなりゲームに近づいてきましたが、あの画面にたとえ敵が出てきたとしても、それだけではゲームとして全く成立しません。リアルタイム・ゲームがゲームとして成立するには、敵の動きと主人公との間に何らかのコミュニケーションがなければならぬからです。それでは、敵と主人公とのコミュニケーションとは何かといえ、結局は衝突の判定ということになります？ 衝突とは、2つのパターンが接触、あるいは重なっているかどうかですから、その判定はそれぞれの座標を調べれば簡単にわかるわけです。これを、具体的な形で示すと図1のようになります。

この図でいう接触の判定とは、2つのパ

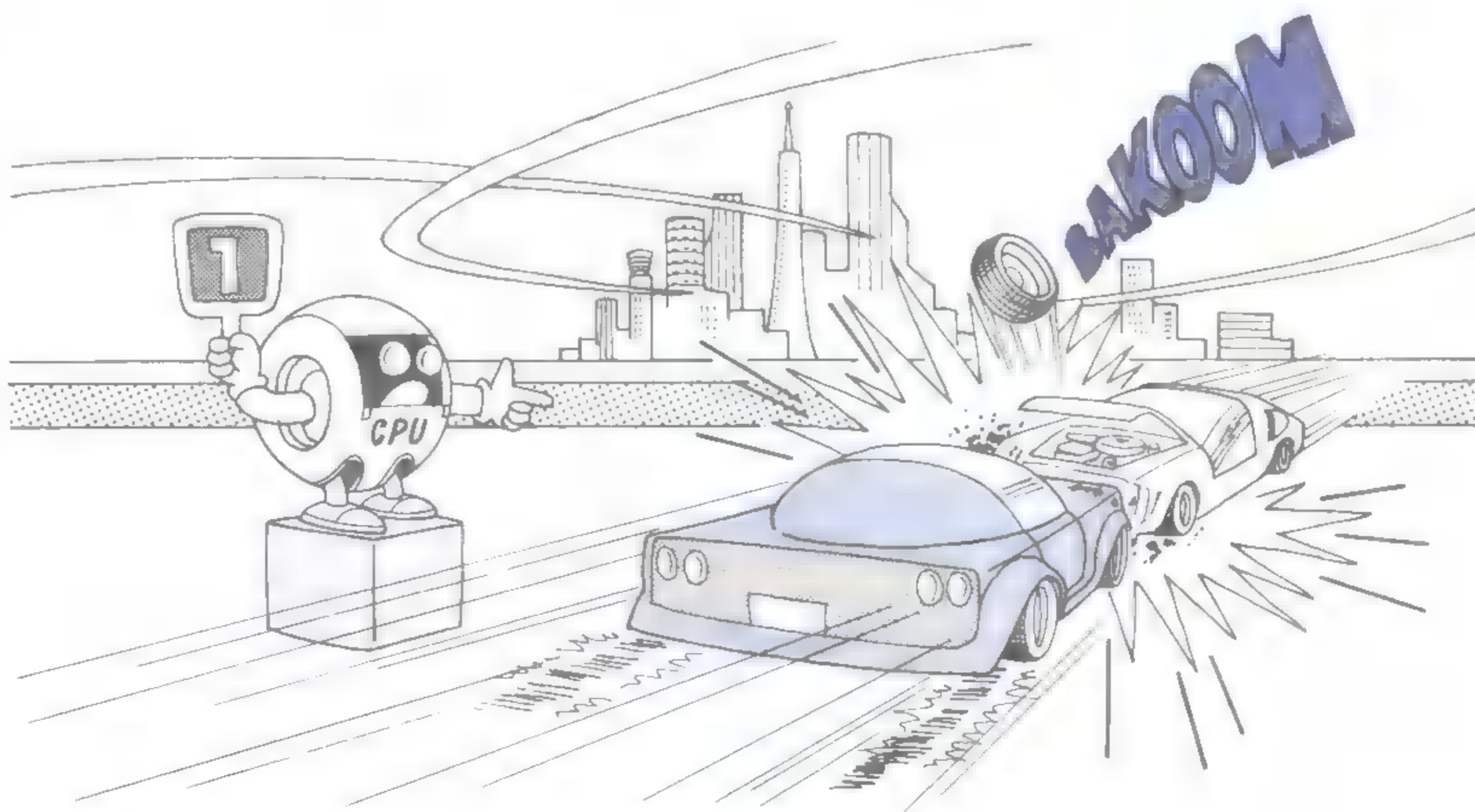
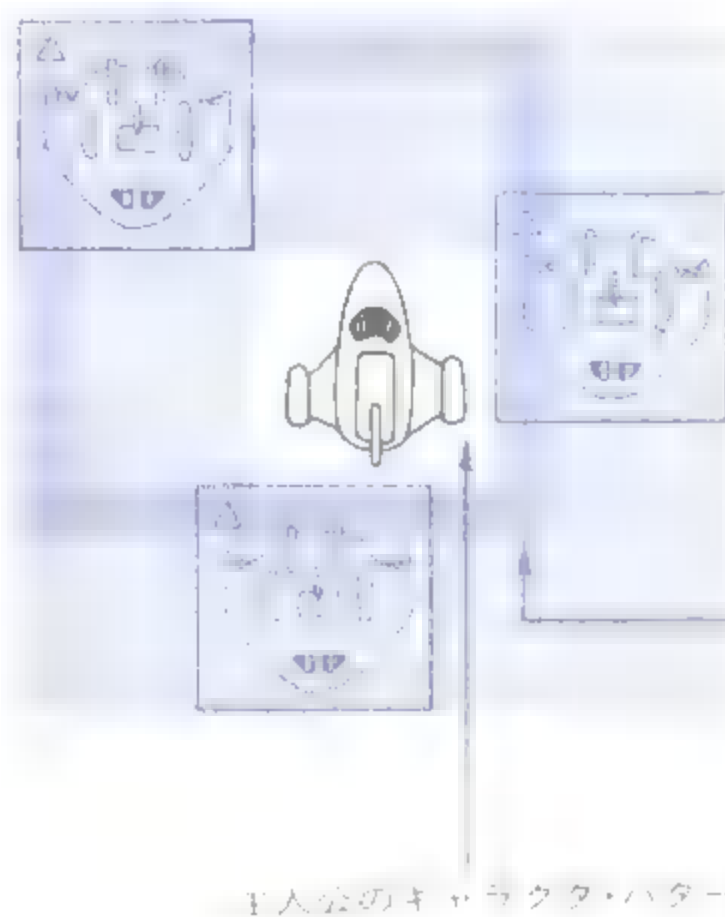
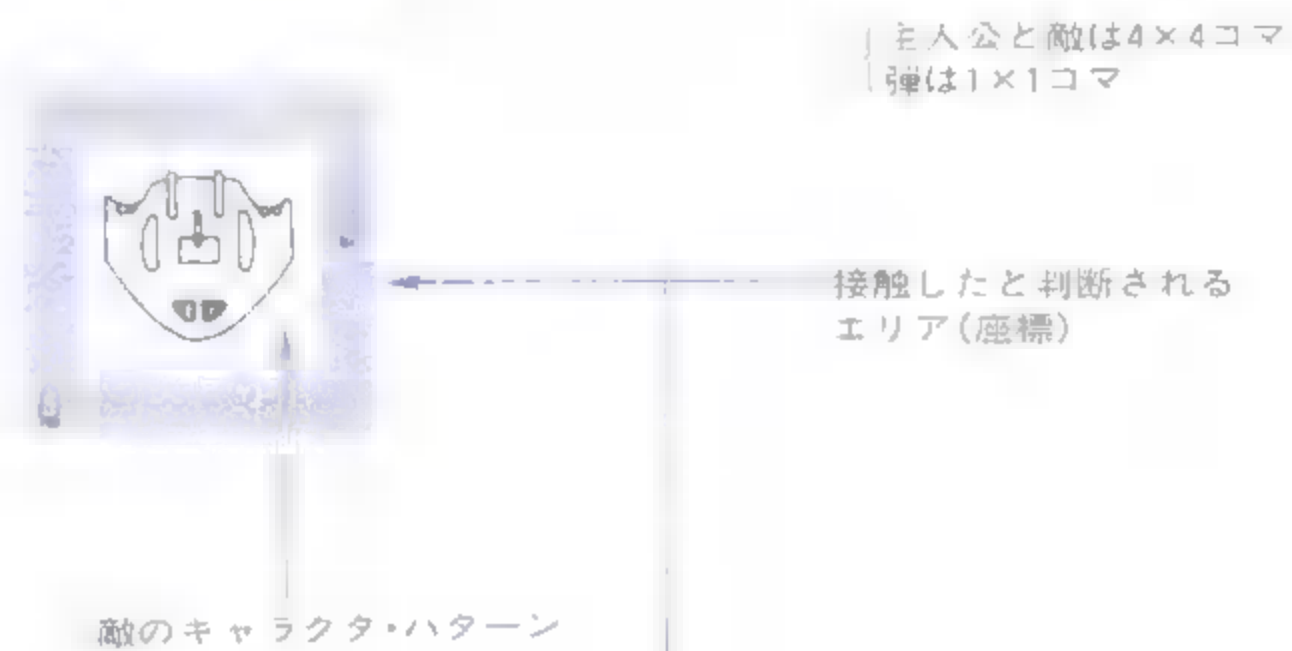


図 1

1. 主人公と敵の接触



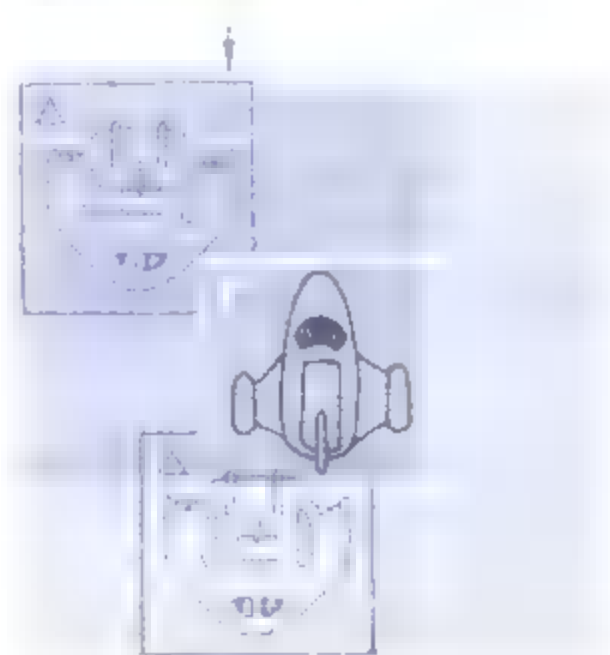
2. 弾と敵の接触



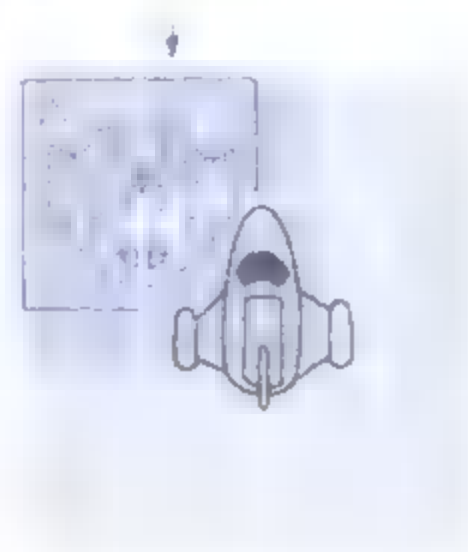
★…接触もしくは衝突の判断をするために基準となるハターンの先頭座標
 ……接触もしくは衝突を判断されるハターンの先頭座標の例

衝突したと判断される座標

3. 主人公と敵の衝突 I



4. 主人公と敵の衝突 II



5. 弾と敵の衝突



ターンが出会った時に互いにハジキ合うというような場合(ピンボールゲームなど)用いられます。一方、衝突というのは2つのパターンがある程度、重なった場合をいいます。どちらにしても判定の基準は図1で示されるようにパターン左上(先頭座標)の位置関係になります。

衝突1の判定は、敵と主人公が1コマでも重なると、衝突したとみなす判定法です。キャラクタ・パターンは、4×4ブロックで構成されていますが、その全部にパターンが描かれているのではなく、空白部分も含んでいます。そのため、1コマ重なるだけで衝突したと判定されたのでは、描かれた図柄自体は \blacksquare になっていない場合があります。ゲームとしてはキビシ過ぎるといえます。そこで衝突2のように、パターンの 1_8 (2コマ)以上が重なった時、初めて衝突したと判定することになっています。なお、弾と敵の衝突については、ややキビシク敵のパターン

(4×4ブロック)内に弾があれば衝突の判定をすることになっています。ゲームをするあなたにとってやや有利であるといえます。

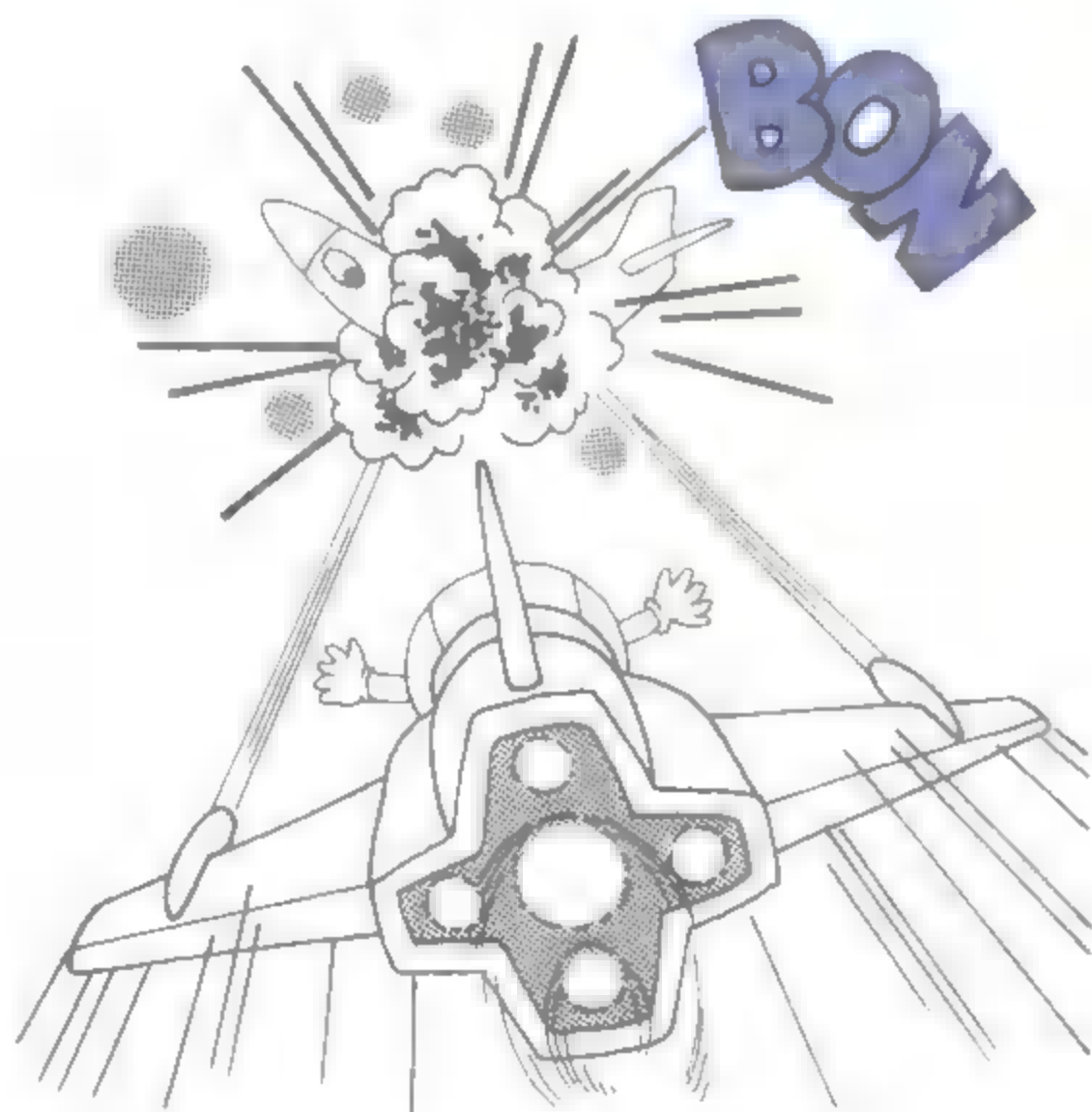
このList 3-1のMYCHKルーチン(26020~26270行)が、敵と主人公との衝突判定をするルーチンで、EMCHKルーチン(26290~26460行)は敵と弾との衝突判定をしています。判定の結果はどちらもキャリーフラグで返していますから、メイン・ルーチン(List 3-5のTESTルーチン)ではこの判定プログラムの後で、キャリーフラグによる条件分岐をさせれば、衝突後の処理ができることになります。この2つのルーチンを見ると、EMCHKの方はシンプルでわかりやすいと思うのですが、MYCHKの方の衝突座標の計算方法は、何だかわかりにくいような気がするかもしれません。2を足して5と比較するなら、何も足さないで3と比較しても同じではないか、と思いたくなりますね。これは、X、Y共に同じことなのですが、比較判定を1回で済ますのに、式を次のように直しているためです。

《衝突と判定される位置関係》

$$-2 \leq \text{主人公の座標} - \text{敵の座標} \leq 2$$

$$0 \leq \text{主人公の座標} - \text{敵の座標} + 2 \leq 4$$

つまり、1バイトの16進数では $-2 = \text{FEH}$ となってしまうので、このままでは $-2 > 3$ という判定をされるのです。これを避けるために、 $+2$ をしてから5と比較するといふのです。どうしてもわかりにくい場合は、両者の座標に数値を入れて計算してみるとハッキリします。



List 3-1 衝突の判定

実行不用…このプログラムには TEST ルーチンがないため

```

;***** List 3-1-N *****
;
C24A MYCHK: ;MY Check
C24A 216EC1 LD HL,EMWORK
C24D 110E00 LD DE,14
C250 0603 LD B,EMVAL
C252 MCLOOP: ;My Check Loop
C252 7E LD A,(HL)
C253 23 INC HL
C254 23 INC HL
C255 B7 OR A
C256 2815 JR Z,NCRASH
C258 3A36C2 LD A,(MYLOC)
C25B 96 SUB (HL)
C25C C602 ADD A,2
C25E FE05 CP 5
C260 300B JR NC,NCRASH
C262 3A37C2 LD A,(MYLOC+1)
C265 23 INC HL
C266 96 SUB (HL)
C267 2B DEC HL
C268 C602 ADD A,2
C26A FE05 CP 5
C26C D8 RET C
C26D NCRASH: ;No CRASH
C26D 19 ADD HL,DE
C26E 10E2 DJNZ MCLOOP
C270 C9 RET

;
C271 EMCHK: ;Enemy Check
C271 2138C2 LD HL,BULWOK
C274 0606 LD B,BULVAL
C276 ECLOOP: ;Enemy Check Loop
C276 7E LD A,(HL)
C277 23 INC HL
C278 B7 OR A
C279 2811 JR Z,NEC
C27B 7E LD A,(HL)
C27C DD9602 SUB (IX+2)
C27F FE04 CP 4
C281 3009 JR NC,NEC
C283 23 INC HL
C284 7E LD A,(HL)
C285 2B DEC HL
C286 DD9603 SUB (IX+3)
C289 FE04 CP 4
C28B D8 RET C
C28C NEC: ;Not Enemy Crash
C28C 23 INC HL
C28D 23 INC HL
C28E 10E6 DJNZ ECLOOP
C290 C9 RET

```

—主人公と敵の衝突判定—

HL ← HL + DE → の値が 0 になり、
 0 になった場合は衝突判定が成立する。
 1 秒には 5 回チェックが行われる。

敵が弾との衝突チェック

HL ← HL + DE → の値が 0 になり、
 0 になった場合は衝突判定が成立する。
 1 秒には 5 回チェックが行われる。

弾の出現

弾の出現座標 - 敵の座標 < 0 なら、
 1 秒には 5 回チェックが行われる。

衝突している

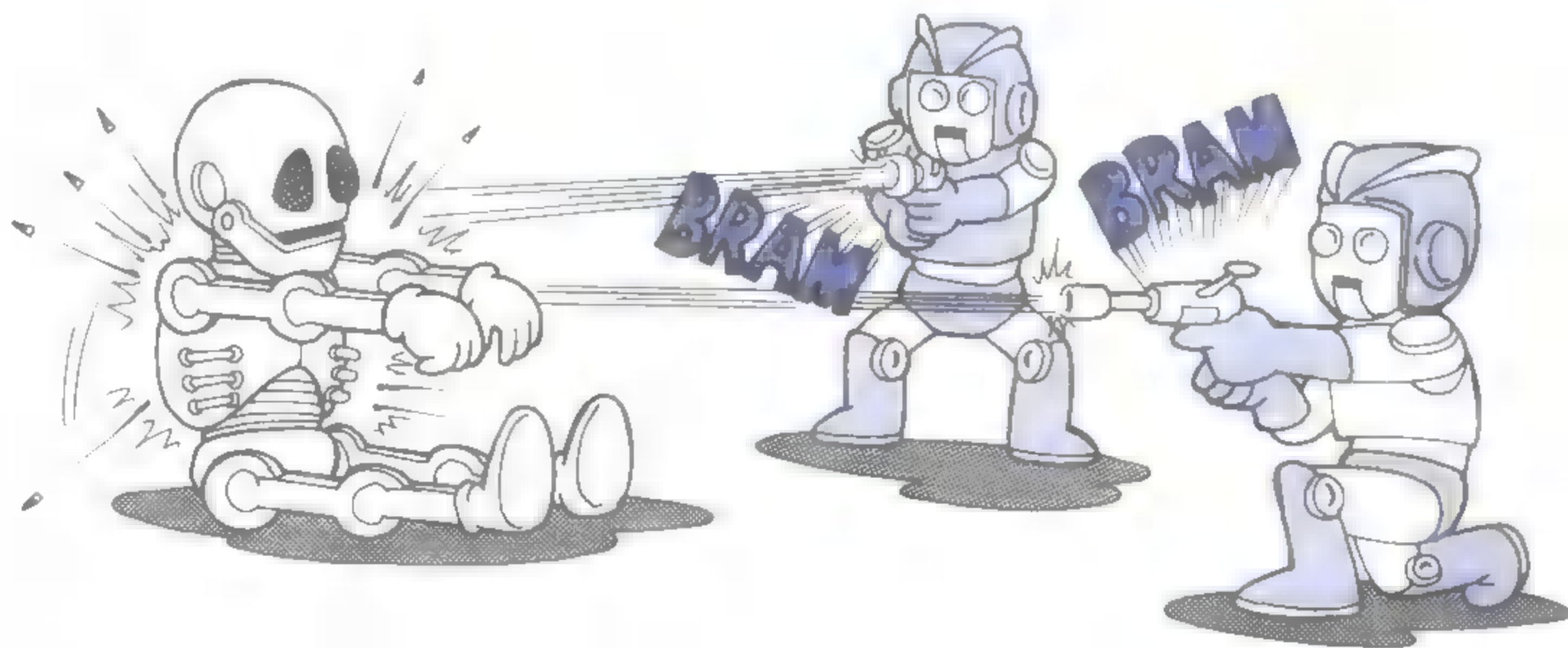
2. 数字…文字と数字パターンの作製

衝突のチェックが済めば、次は後処理をしなければなりません。つまり、裁判でいうなら刑の執行となるか、無罪放免となるかですが、ゲームでは、敵が弾に当たって爆発するとか、点数をアップするとか、主人公がやられたらゲーム・オーバーになるとか…ということになります。ゲームでは、死んでもすぐに生き返れるので、死への恐怖などというものは誰も感じないと思います。しかし、元来人間にとってこの恐怖はとても大きなものであったのです。そのために生まれたのが宗教というものであり、キリスト様も、お釈迦様も、アラールの神も……。すべてこの死への恐怖を取り除いてくれる(?)という点で一致しているのです。そういう意味においては、コンピュータ・ゲームは正に時代の■先端を行く宗教といえよう…?!


ここでは、そのような恐怖(?)処理の準備として、数字や文字を表示するためのルー

チンを作成します。数字や文字を表示するルーチンといっても、基本的な画面表示の考え方は、パターン表示ルーチンと特に変わりはありません。ただ、文字パターンのサイズ、色(ここでは白に統一)、連続表示、などの点から、これまでのパターン表示プログラムに少し変化があるという程度です。

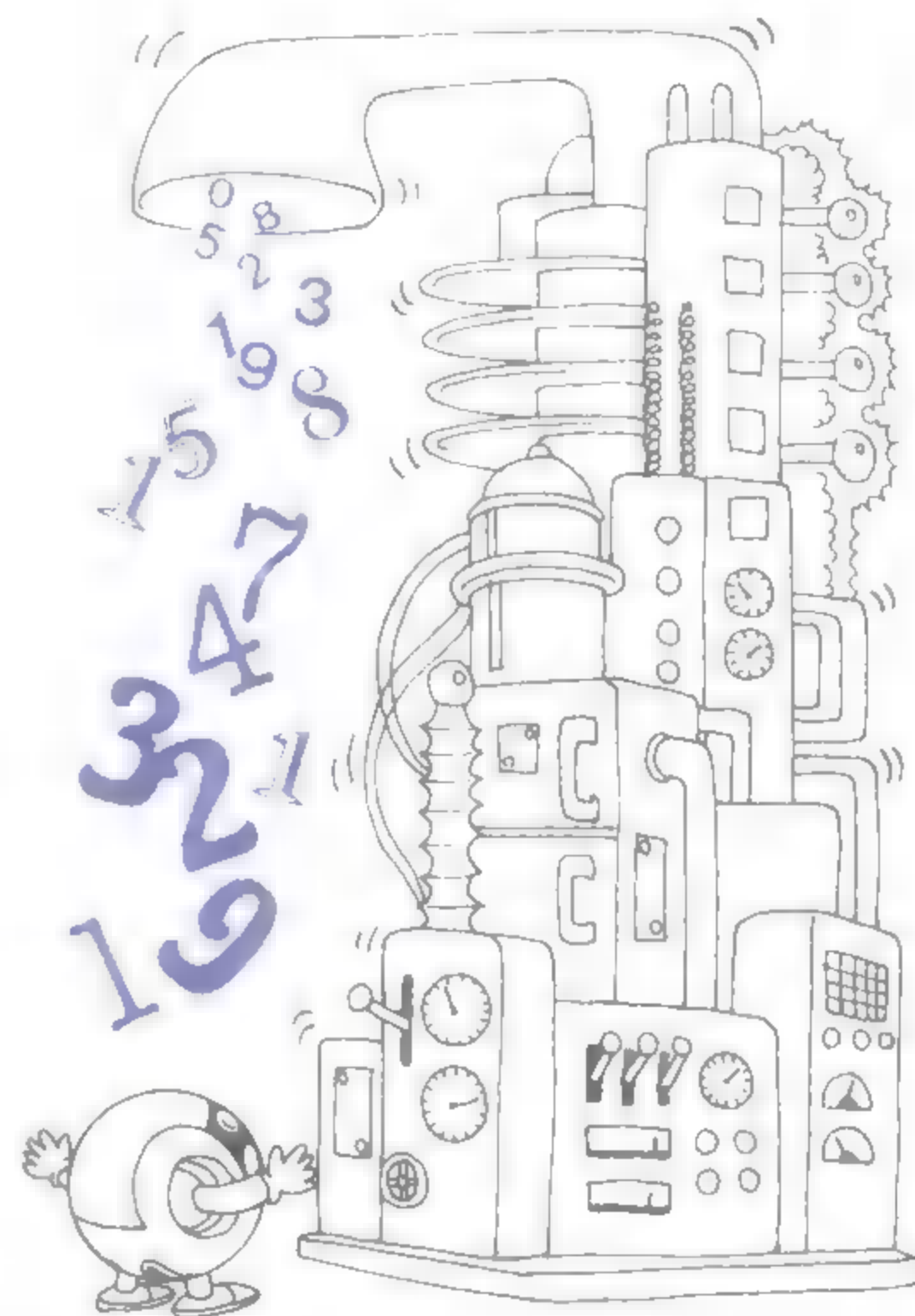
そこで、まず必要になってくるのは数字や文字のグラフィック・データです。これがないと表示プログラムが正しいかどうかの実験もできませんから、ここはめんどくでもすべての数字、文字用グラフィック・データをパターン・エディタで作ってしまいます。パターン・サイズは16×8ドットで、連続した時に上下左右が触れないようカラーページの⑤のように最初から一回り小さく作ります。細かいことをいえば、下部のスペースはわざわざデータで持つ必要はないのですが、ここではデータの管理をわか



りやすくするために（データがちょうど 10H バイト単位になる）、あえてデータとしてあります。

グラフィック・データは、1 面分しか必要としないので、パターン作成が終了したら 1 のタイプのデータを選び、さらにチェンジ・データで R と G を削除します。これで、ブルー面 1 面分のデータが B500H~B50FH 番地にできてきますので、カラーページの⑤に示されているアドレスに順次転送をしていきます。なぜブルー面だけのデータでいいかというと、すでに気がついていると思いますが、文字の表示色である白は B・R・G 3 面共同じデータが入っているからです。このことは、1 つのデータがあればプログラムによって、7 色の文字を表示できることにもなります。ここでのプログラムは、白になるようにしかしていませんが、将来オリジナルのゲームを作る時には、カラー・ナンバーの指定で描く面が  択できるようにすると、文字のカラー化ができます。

データの転送先で、数字の“9”と文字“A”との間に“□”が挿入されていますが、このパターンは、パターン・エディタで作らなくてけっこうです。めんどうな人は、ここにすべて 00H を入れておいてください。これは、スペースすなわち 1 文字分の空送りをする時に、消去用のデータとして使っているからです。そして、すべての数字・文字データが揃ったら忘れずにセーブしてください。その際、2 章で作成した各種パターンのデータとまとめておくと、この後グラフィック・データのロードが 1 回で済むようになります。ただし、次に作るゲームで



もこの数字・文字データは使用することになりますので、単独でもセーブしておいた方がいいでしょう。

さて、p.90 の List 3-2 の内容は 1 文字の表示、画面全部の高速消去、連続文字の表示の 3 部から成っています。このプログラムでは、これまでパターンの表示にアドレスを求めるために使っていた、パターン番号からデータ・アドレスを計算するルーチン(PDADR)が使われていません。その代わりに SEEKLD(27040~27140 行)という別の変換ルーチンが出てきています。これは、

特に意味のあることではなく PDADR はパターン・サイズがバラバラでも利用できる例として用い、また今回のようにデータ長が一定の場合には、ポインタをわざわざ確保する必要がないので、別のルーチンにしただけのことです。したがって、数字・文字にはこれまでのパターン番号とは別に、次のようなパターン番号がついていると解釈できます。

パターン番号 00-09 : 数字の 0-9
 パターン番号 10 : スペース
 パターン番号 11-36 : 文字の A-Z

ここで工夫を凝らしているのは数字、文字の連続表示ルーチンで、連続表示データとしてアスキーコードが使えるという点です。これは、数字表示の場合はあまりメリットがありませんが、文字を表わす時には表示したい文字を ` で囲むだけで、連続表示でき文字表示がおおはばに楽になるからです。アスキー・コードから前述のパターン番号に直す分だけ、プログラムとしてはほんの少し長くなりますが、MESS の内容を文字データにしてアセンブルすると、パターン番号でデータを作ることがいかにめんどうか、すぐわかると思います。連続表示の

エンド・サインは 0 となっているので、これは間違っても、` で囲まないようにしてください。

もう 1 つのルーチン、高速画面クリアについては出力ポート 31H がここでのミソになります。このポートへ出力する前に、AND 0F7H を実行していますが、これは特定のビット(ここではビット 3)を 0 にするために使われる命令でしたね。そして、この出力ポート 31H のビット 3 は、画面表示のオン/オフをするためのビットなのです。ですから、消去している最中は画面を消して、その画面消去の作業を見えないようにしているのです。これを利用すると、背景などを画面を消した状態で描き、いきなりパッと表示するということもできます。カッコ良さを追求するには、覚えておきたいテクニックの 1 つです。なお、E6C2H 番地の意味は出力ポート 31H へ出力したデータが入っているワークエリアですが、BASIC のシステム領域で用いているからです。マシンの語によって操作しても、その値の変化はありません。画面表示をもどす際に、OR 08H をする必要がないのはそのためです。

List 3-2 文字の表示

```

13000          ;***** List 3-2-G *****
              ;
BE92          DISPLE: ;DISPlay Letter
BE92 CD00C0    CALL XYADR
BE95 CD91C2    CALL SEEKLD
BE98 D35C      OUT (5CH),A
BE9A CDB0BE    CALL BOXL
BE9D 2ACBBE    LD HL,(LOADR)
BEA0 D35D      OUT (5DH),A
13090 BEA2 CDB0BE CALL BOXL

```

——文字・数字の表示
 表示アドレスを定めるため
 アドレスを定めるため
 文字・数字の表示


```

BEA5 2ACBBE      LD    HL,(LOADR)
BEA8 D35E        OUT   (5EH),A
BEAA CDB0BE      CALL  BOXL
BEAD D35F        OUT   (5FH),A
BEAF C9          RET

;
BOXL: ;BOX of Letter      1文字の表示
      LD    (LETSP+1),SP
      LD    SP,HLEN-2
      LD    DE,(DISPAD)
      LD    BC,8FFH
      LLOOP: ;Letter LOOP
      LD
      LD
      EX    DE,HL
      ADD   HL,SP
      EX    DE,HL
      DJNZ  LLOOP
      LETSP: ;LETter Stack Pointer
      LD    SP,0000
      RET

;
LOADR: ;Letter Data ADdRes —文字・数字のデータ・アドレスが入る
      DS    2
      ;ワークエリアを確保

E6C2      PORT31:EQU 0E6C2H ;data or output PORT 31h 画面
      ;表示のオン・オフ

;
CLS: ;CLear Screen      画面の高速消去
      LD    A,(PORT31)
      AND   0F7H
      OUT   (31H),A
      OUT   (5CH),A
      CALL  ACLS
      OUT   (5DH),A
      CALL  ACLS
      OUT   (5EH),A
      CALL  ACLS
      OUT   (5FH),A
      LD    A,(PORT31)
      OUT   (31H),A
      RET
      ACLS: ;All CLS
      LD    HL,VTOP
      LD    DE,VTOP+1
      LD    BC,3E7FH
      LD    (HL),0
      LDIR
      RET

;
***** List 3-2-N *****
;
BB00      LBASE: EQU 0BB00H ;Letter BASE address
;
C291      SEEKLD: ;SEEK Letter Data —文字・数字のパターン・データのある
      ADD   A,A      先頭アドレス
C291 87
C292 87

```

27070	C293 6F	LD L,A	
	C294 2600	LD H,0	
	C296 29	ADD HL,HL	
	C297 29	ADD HL,HL	
	C298 1100BB	LD DE,LBASE	
	C29B 19	ADD HL,DE	
	C29C 22C8BE	LD (LOADR1,HL	
	C29F C9	RET	
	C2A0	MSGPRN: ;MeSsaGe PRiNt	---文字列の表示
	C2A0 7E	LD A,(HL)	
	C2A1 B7	OR A	
	C2A2 C8	RET Z	
	C2A3 FE20	CP	
	C2A5 2002	JR NZ,MSG2	
	C2A7 3E3A	LD A,' ' +10	
	C2A9	MSG2: ;MeSsaGe print-2	
	C2A9 D630	SUB '0'	
	C2AB FE0B	CP 11	
	C2AD 3802	JR C,MSG1	
	C2AF D606	SUB 6	
	C2B1	MSG1: ;MeSsaGe print-1	
	C2B1 C5	PUSH BC	
	C2B2 E5	PUSH HL	
	C2B3 CD92BE	CALL DISPLE	
	C2B6 E1	POP HL	
	C2B7 C1	POP BC	
	C2B8 0C	INC C	
	C2B9 0C	INC C	
	C2BA 23	INC HL	
	C2BB 18E3	JR MSGPRN	
27830			
50000		***** List 3-2-T *****	
	D000	TEST: ;TEST	
	D000 F3	DI	
	D001 3100B6	LD SP,STACK	
	D004 CDCDBE	CALL CLS	
	D007 011010	LD BC,1010H	
	D00A 2112D0	LD HL,MESS	
	D00D CDA0C2	CALL MSGPRN	
	D010 FB	EI	
	D011 FF	RST 38H	
	D012	MESS: ;MESSaGe	
50130	D012 30313233	DB '0123456789',0	
	D016 34353637		
	D01A 383900		

3. 計算…得点の計算と表示 その1

数字や文字を自由に画面に表示できるようになったからといって、すぐに得点の表示が可能になったのではありません。得点を画面に表示するには、まだ重要な問題が残っています。問題は、コンピュータが計算するのは16進数で画面に表示するのは10進数であるという点です。つまり、内部では16進数で計算をしていますが、人間には10進数表記でないと理解されないということです。たとえば、マシン語でプログラムを組むことができるような人でも、16進数より10進数の方がわかりやすいのは当然のことです。このあたりのギャップが、人間の頭脳とコンピュータとの基本的な構造上の違いであるといえます。

この責任のすべては、人間を創造した神様にあります。もしも、人間の指が8本ずつあったならば、最初からすべて16進数の世界になっていたはずですよ。おそらく神様も人間がこのような奇怪な機械を、創造するとは夢にも思わなかったのでしょう。コンピュータの出現に一番驚いたのは、そういう意味では人間を創造した神様であったかもしれません。しかし、今後は人工知能を持ったコンピュータが、さらに知恵を持つ何かを創造するようになると、コンピュータにとっての神である人間が、その違いに驚くという時が来るかもしれません。

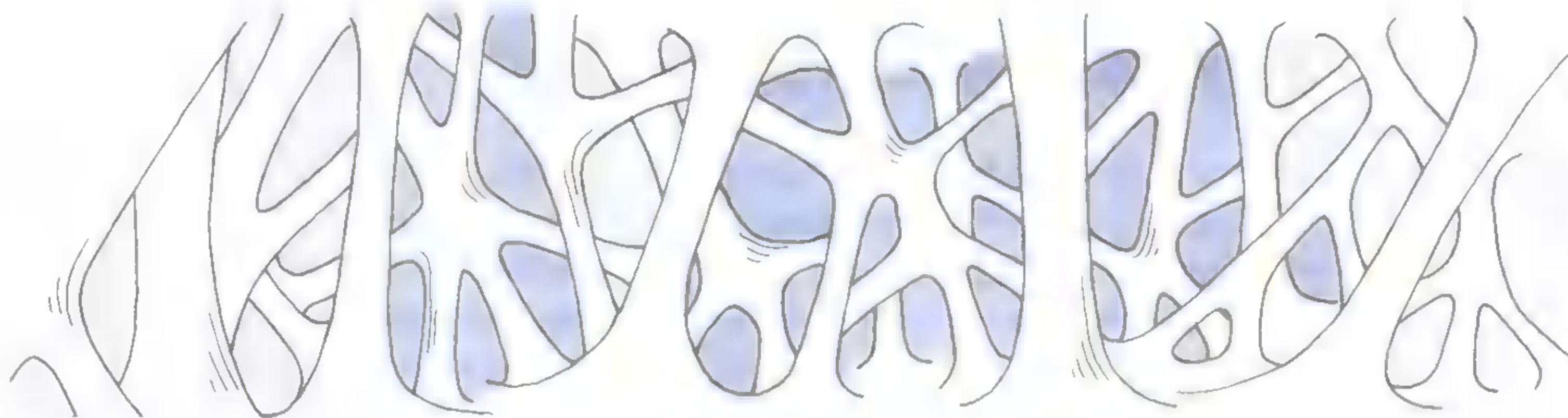
ここでは、16進数の数字データを10進

数の連続文字(数字)列に変換することで、この問題を解決していきます。0から9までの数字の列になれば、作成したばかりの文字、数字連続表示ルーチンを使って、画面に出力できるようになるのです。

マシン語で直接計算できる数字は0-FFFFHまでですから、スコア専用のワークエリアとして、2バイトのメモリを用意する必要があります。これは、10進数でいうと0-65535にしかありませんが、実際にはスコア表示の際は、ダミーとして最後に00をつけておくことにより0-6553500という高い点数にすることができます。これだけの点数があれば、表示スコア不足になることはまずありません。ダミーとわかっているとしても、この00がないと不満があるというのは、ゲームの世界も相当にインフレが進んでいるからでしょう。

p.95 の List 3-3 は、このような2バイトの16進数からなるスコアを、DEレジスタで示される数字と加算した上で、10進数の文字列に変換し、指定位置から表示するというプログラムです。

リストのコメントを見れば明らかなように16進数から10進数への変換は求める桁ごとに割算をして、その桁の数字を出しています。この変換計算の考え方は、次のように10進数の数字でやってみると理解しやすくなります。



例 65535 (FFFFH) の各桁の値を求める

1. 10000 (2710H) で 65535 (FFFFH) を割る

商……6——10000 の位

余り…5535 (159FH)

2. 1000 (3E8H) で 5535 (159FH) を割る

商……5——1000 の位

余り…535 (217H)

3. 100 (64H) で 535 (217H) を割る

商……5——100 の位

余り…35 (23H)

4. 10 (0AH) で 35 (23H) を割る

商……3——10 の位

余り…5——1 の位

結局、割る数も割られる数も 16 進数であれば、各桁の値は同じように求められるのです。そして、この各桁の値に 30H を加えることにより、この数字がアスキー・コードとなります。これらを、順次指定のメモリーに格納することによって、それらはそのまま連続表示用のデータとなるので、データ終了のサイン 00 を最初から 1 の位の次のメモリーに入れておけば OK です。

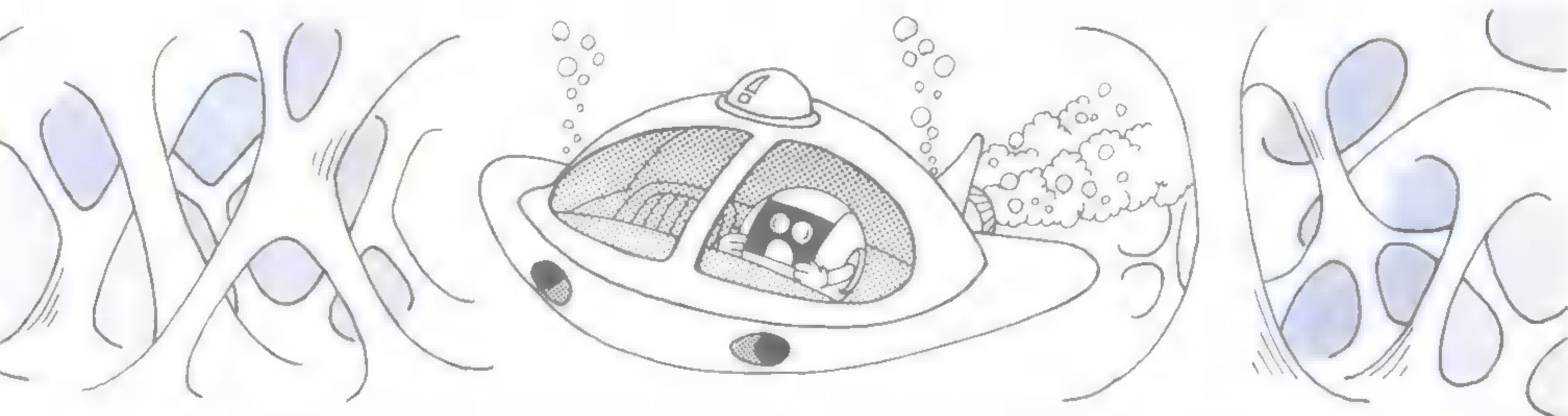
なお、ここでの割算というのは説明上のことで、実際には引き算を何度も繰り返し、

引いた回数をカウントして商を求めています。そして、最後には引き過ぎとなるので、引き過ぎた分だけ足して余りとしています。

では、実際にスコアを表示させるテストをしてみましょう。表示させる内容は次のように指定します。テストとはいえ、ダミーの 00 までついた立派なものです。

BC レジスタ：10000の位の表示座標 (C, B)

DE レジスタ：加算するスコア



いつものように D000H 番地から走らせると、スコアが100点(実際は1点)ずつアップしていくはずですが。といっても、あまりに高速で1/100秒計時のストップ・ウォッチのように見えるかもしれません。このテストには、終わりが無いので適当にストップキーを押して止めてください。

もし、画面数などで2桁の数字を表示し

たい場合は、連続表示データのスタートを現在の F10000 から F10 に変更すれば、2桁の表示に変更することができます。ところで、せっかく理解してきたこの得点表示プログラムなのですが、実は本書ではこのテスト以外に使われることのない、幻のプログラムになる運命なのです…。

List 3-3 得点の計算と表示(その1)

```

;***** List 3-3-N *****
;
DSC1:
C2BD          PUSH BC
C2BD C5        LD HL,(SCORE)
C2BE 2A09C3    ADD HL,DE
C2C1 19        LD (SCORE),HL
C2C2 2209C3    ;
C2C5 011027    LD BC,10000
C2C8 CDF7C2    CALL DIVIDE
C2CB 3203C3    LD (F10000),A
C2CE 01E803    LD BC,1000
C2D1 CDF7C2    CALL DIVIDE
C2D4 3204C3    LD (F1000),A
C2D7 016400    LD BC,100
C2DA CDF7C2    CALL DIVIDE
C2DD 3205C3    LD (F100),A
C2E0 010A00    LD BC,10
C2E3 CDF7C2    CALL DIVIDE
C2E6 3206C3    LD (F10),A
C2E9 7D        LD A,L
C2EA C630      ADD A,30H
C2EC 3207C3    LD (F1),A

```

28230					
C2EF	C1		POP	BC	BC の値をスタックから取り出す
C2F0	2103C3		LD	HL,F10000	HL ← F10000 文字列の先頭アドレス
C2F3	CDA0C2		CALL	MSGPRN	(C,B) より文字列を表示するため
C2F6	C9		RET		
C2F7					
		DIVIDE:		DIVIDE HL by BC	
C2F7	AF		XOR	A	A ← 0, 同時にキャリーフラグのリセット
C2F8		SBMORE:		SuBtract MORE	
C2F8	ED42		SBC	HL,BC	HL ← HL - BC
C2FA	3803		JR	C,OVER	HL < 0 なら OVER
C2FC	3C		INC	A	A ← A + 1
C2FD	18F9		JR	SBMORE	ジャンプ
C2FF		OVER:		SuBtract OVER	
C2FF	09		ADD	HL,BC	HL ← HL + BC
C300	C630		ADD	A,30H	A ← A + 30H
C302	C9		RET		
C303		F10000:DS	1	;Figure 10000	10000 の位の値がアスキーコードで入る
C304		F1000:DS	1	;Figure 1000	1000 の位の値がアスキーコードで入る
C305		F100:DS	1	;Figure 100	100 の位の値がアスキーコードで入る
C306		F10:DS	1	;Figure 10	10 の位の値がアスキーコードで入る
C307		F1:DS	1	;Figure 1	1 の位の値がアスキーコードで入る
C308		FEND:DS	1	;Figure END	文字列のエント・サイン 0 が入る
C309		SCORE:DS	2	;SCORE	
28481					
50000					
				***** List 3-3-T *****	
D000		TEST:		TEST	
D000	F3		DI		
D001	AF		XOR	A	
D002	D351		OUT	(51H),A	DMA をオフにするため
D004	3100B6		LD	SP,STACK	スタックポインタを B600H 番地に設定
D007	CDCDBE		CALL	CLS	画面をクリア
D00A	AF		XOR	A	A ← 0
D00B	3208C3		LD	(FEND),A	
D00E	3E30		LD	A,30H	A ← 30H → 0
D010	3206C3		LD	(F10),A	
D013	3207C3		LD	(F1),A	
D016	210000		LD	HL,0	
D019	2209C3		LD	(SCORE),HL	
D01C	014A00		LD	BC,004AH	BC ← マルチスコア「00」の表示位置
D01F	2106C3		LD	HL,F10	
D022	CDA0C2		CALL	MSGPRN	(C,B) より文字列を表示するため
D025		TLOOP:		Test LOOP	
D025	110100		LD	DE,1	DE ← 1 加算スコア
D028	014000		LD	BC,0040H	BC ← 0040H 表示座標
D028	CDBDC2		CALL	DSC1	
D02E	DB09		IN	A,(9)	スコア (9) を加算して (C,B) より表示
D030	1F		RRA		
D031	38F2		JR	C,TLOOP	キーが押されていない場合は「LOOP」へ
D033	FB		EI		
50260	D034		RST	38H	

4. BCD…得点の計算と表示 その2

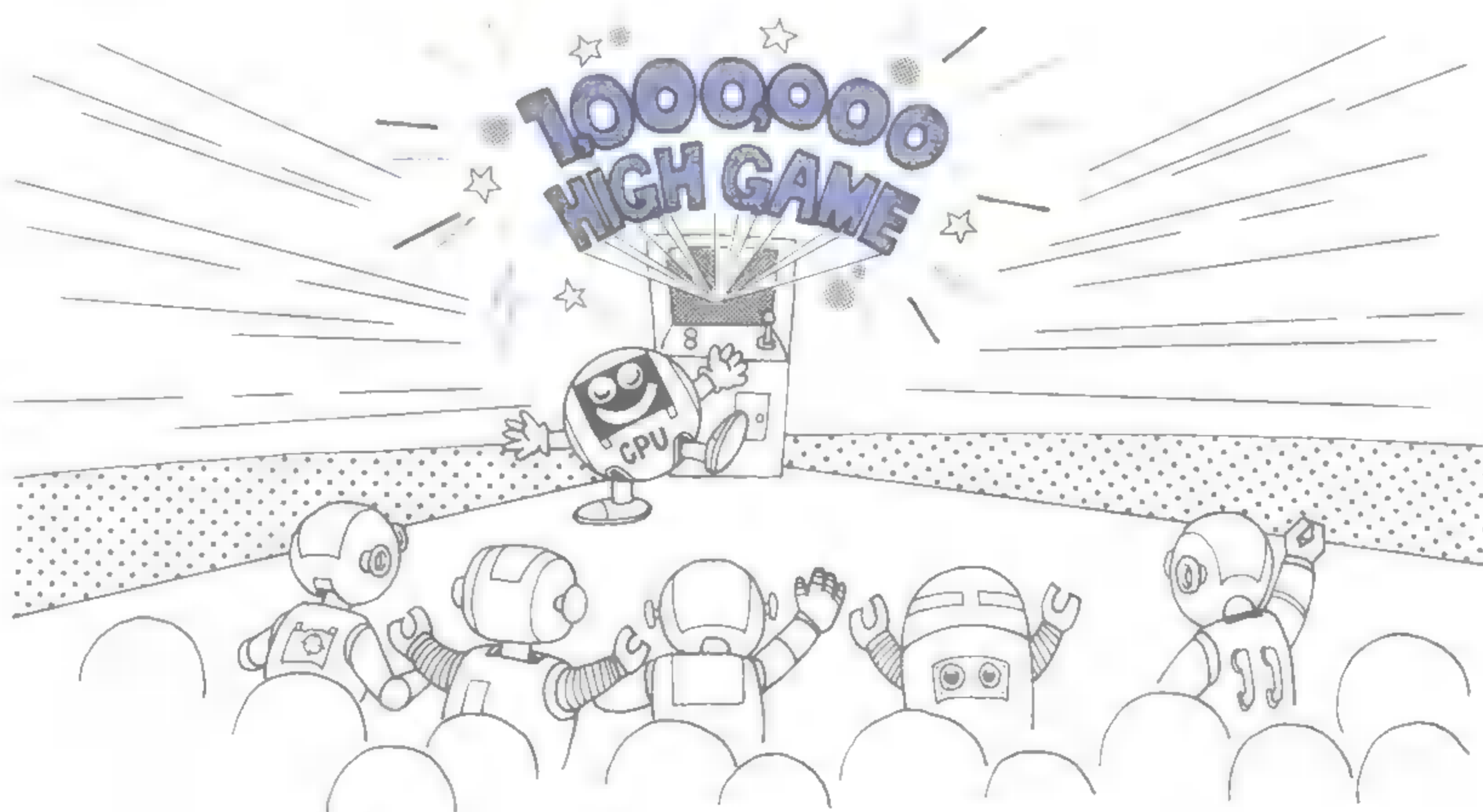
普通のゲームであれば、16進数→10進数への換算による得点表示でもまったく支障はありませんし、困ることはありません。しかし、得点を表示するたびにイチイチ換算するというのは、どう考えても合理的な方法であるとはいえません。それに、たとえ不足することがないといっても、数値の上限に最初から制限があるというのもあまり気分のいいものではありません。この2つの不満を、一気に解消するようなウマイ方法はないのでしょうか……

実は、この16進数と10進数との問題は、ゲームばかりではなくコンピュータと人間がコミュニケーションする上で、常に存在している大きな障害なのです。例えば、電卓な

どは計算がすべてという商品ですから、《入力は10進数、内部計算は16進数、表示は10進数》ではたまりません。そこで、16進数の内AH-FHまでは使用しないで、16進数を10進数の感覚で使ってしまうというのが、BCD(Binary Coded Decimal)=2進化10進数の考え方なのです。

これは何を意味するかというと、使う側が0から9までの数字だけを使い、16進数を10進数とみなしてしまおうというものです。ですから、計算をしない限りは10進数そのものとまったく同じことなのです。

例 16進数で12=10進数で12とみなす
16進数で15=10進数で15とみなす



このように、16進数を10進数と同じものと考えすることは、考える人の勝手ということになりますが、ここに計算処理が入るとそう単純にはいかなくなります。つまり、コンピュータはあくまで16進数しか処理できないからです。そのために、計算をした時にはかならず16進(2進)から10進への補正処理をする必要がでてきます。

例 $12H + 3H = 15H \cdots$ (補正) $\rightarrow 15$
 ……そのままが良い
 $12H + 9H = 1BH \cdots$ (補正) $\rightarrow 21$
 $18H + 8H = 20H \cdots$ (補正) $\rightarrow 26$
 $37H - 9H = 2EH \cdots$ (補正) $\rightarrow 28$

この16進(2進)から10進への補正は、計算をするたびにかならずしなければなりません。しかし、この補正はDAA(Decimal Adjust Accumulator)というたった1つの命令で解決できるようになっているのです。DAAを実行することにより、アキュムレータの値は10進数に補正されます。さらに、計算の結果で2桁を越える桁上がり、または減

算でマイナスになる時には、キャリーフラグがセットされて対応します。

例 $99H + 99H = 32H$ CF=1…(DAA)
 $\rightarrow 98$ CF=1
 $50H + 50H = ADH$ CF=0…(DAA)
 $\rightarrow 00$ CF=1
 $25H - 30H = F5H$ CF=1…(DAA)
 $\rightarrow 95$ CF=1

このBCDによる計算は、あくまでも使う側が0-99の数字だけで計算をさせないとまったく意味がなくなりますので、勘違いしてはいけません。つまり、計算する数に、1DHとか4BHなどの10進数にはない数字が含まれていた場合には、無意味な計算に終わってしまいます。

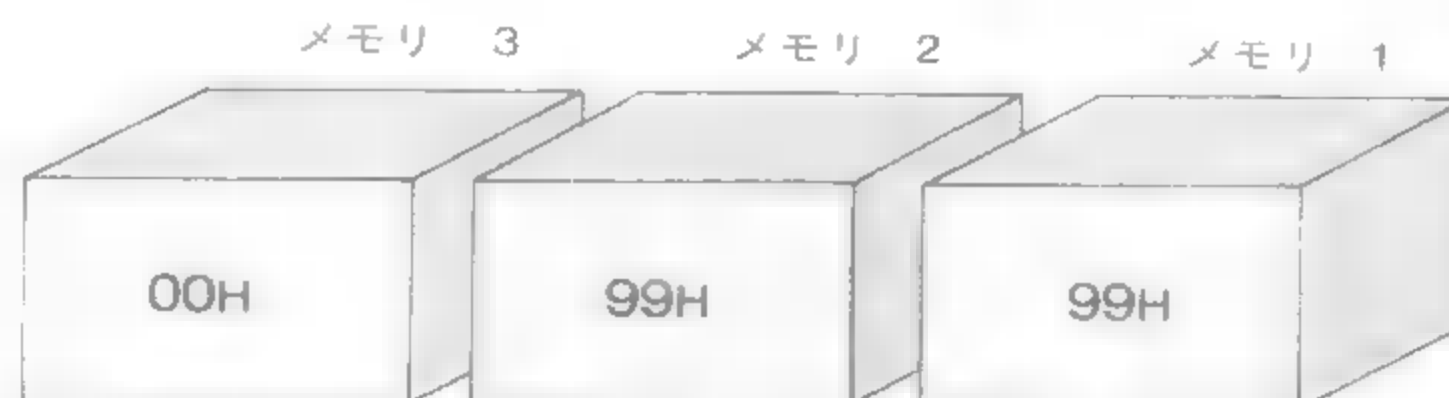
では、このBCDで計算させると、桁数はどこまで取れるのでしょうか。結論からいうと、これはメモリのあり限り無限に取れることになります。その理由は、次の例を見ればわかるでしょう。

図 2

例 9999Hに99Hを加える場合(桁数は6桁とする)

■ 6桁なので頭に00を付ける

00 99 99H



■ 2桁ずつ(1バイト)メモリに記憶される

[1]

	00H	99H	99H
+		1	99H
<hr/>			
			32H

足した結果のキャリーフラグ(CY)

ここで DAA を行なうと、演算結果の 32H が 98H に補正され CY が立つ

	00H	99H	99H
+		1	99H
<hr/>			
			98H

補正結果

次の 2 桁と CY を加える

[2]

	00H	99H	99H
+	0	1	99H
<hr/>			
		9AH	98H

足した結果のキャリーフラグ(CY)

ここで DAA を行なうと 9AH は 00H に補正され、CY が立つ

	00H	99H	99H
+	1	1	99H
<hr/>			
		00H	98H

補正結果

最後に最上位 2 桁と CY を加える

[3]

	00H	99H	99H
+	0	1	99H
<hr/>			
	01H	00H	98H

足した結果のキャリーフラグ(CY)

結果を DAA する

	00H	99H	99H
+	0		99H
<hr/>			
	01H	00H	98H

補正の必要がないので変わらない

桁数を増やすには、このように、必要な桁数分これらの操作を繰り返していけば良いので理論上は無制限の桁数を演算することができます。これをプログラミングしたのが List 3-4 ですが、ここでは桁数の上限を 6 桁とし、加算する数は 4 桁(DE レジスタで示される数字)まで指定できるようにしてあります。これにダミーの 00 をつければ、00-99999900 までの表示ができるということになります。このようにして計算された数字は、数字の列としての連続表示ではなく、そのまま上位ビットと下位ビットとの数字に分割した上で、1 文字表示のルーチンに飛ばします。SCOREP(29260～29440 行)の所で、ローテイトしたり AND をとっているのはその分割をしているためです。

例 15 を表示する場合

RLCA を 4 回繰りかえす…A=51 となる

0FH との AND をとる…A=01 となる

→ 1 を表示

A に再び 15 を入れる

0FH との AND をとる…A=05 となる

→ 5 を表示

テスト・プログラムの内容は、List 3-3 とまったく同じことをしていますので、実験をしてみてください。考え方さえ理解できれば、BCD の方が制限がないだけ使いやすいともいえます。本書では、約束通り(?)こちらの方を採用していくことになっています。

List 3-4 画点の計算と表示(その2)

```

;***** List 3-4-N *****
;
003E      SCLOC: EQU 003EH ;Score Location —スコア表示場所
;
C30B      DISPSC: ;Display Score —スコアの表示
C30B 013E00      LD BC,SCLOC
C30E 213FC3      LD HL,SCOREL
C311 7B          LD A,E
C312 86          ADD A,(HL)
C313 27          DAA
C314 77          LD (HL),A
C315 2B          DEC HL
C316 7A          LD A,D
C317 8E          ADC A,(HL)
C318 27          DAA
C319 77          LD (HL),A
C31A 2B          DEC HL
C31B 3E00        LD A,0
C31D 8E          ADC A,(HL)
C31E 27          DAA
C31F 77          LD (HL),A
C320 CD28C3      CALL SCOREP
C323 23          INC HL
C324 CD28C3      CALL SCOREP
C327 23          INC HL
;
C328      SCOREP: ;SCORE Print
C328 7E          LD A,(HL)
C329 07          RLCA
C32A 07          RLCA
C32B 07          RLCA
C32C 07          RLCA
C32D CD31C3      CALL PRINTF
C330 7E          LD A,(HL)
;
C331      PRINTF: ;PRINT Figure
C331 E60F        AND 0FH
C333 C5          PUSH BC
C334 E5          PUSH HL
C335 CD92BE      CALL DISPLE

```



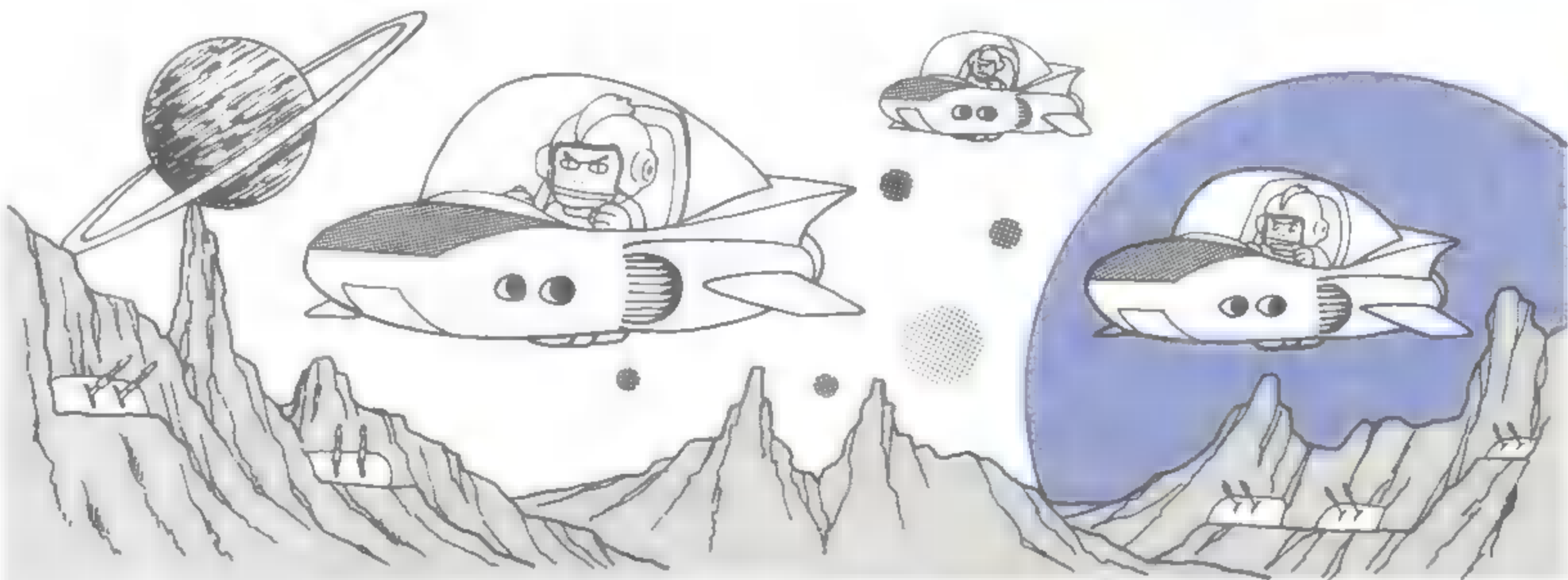
```

C338 E1      POP HL
C339 C1      POP BC
C33A 0C      INC C
C33B 0C      INC C
C33C C9      RET

;
C33D          SCORE2: ;SCORE
C33D          DS 1
C33E          SCORE1: ;SCORE 1
C33E          DS 1
C33F          SCOREL: ;SCORE
C33F          DS 1
C340          DUMMY: ;DUMMY
C340 303000   DB '00',0

;
;***** List 3-4-T *****
;
D000          TEST: ;TEST
D000 F3      DI
D001 3100B6   LD SP,STACK
D004 AF      XOR A
D005 D351     OUT (51H),A
D007 323FC3   LD (SCOREL),A
D00A 323EC3   LD (SCORE1),A
D00D 323DC3   LD (SCORE2),A
D010 CDCDBE   CALL CLS
D013 014A00   LD BC,004AH
D016 2140C3   LD HL,DUMMY
D019 C0A0C2   CALL MSGPRN
D01C          TLOOP: ;Test LOOP
D01C 110100   LD DE,1
D01F CD0BC3   CALL DISPSC
D022 DB09     IN A,(9)
D024 1F      RRA
D025 38F5     JR C,TLOOP
D027 FB      EI
D028 FF      RST 38H

```



5. 衝突の処理…ゲームらしさの追求

衝突の判定、得点の表示が可能になれば、最後の仕上げとして全体をまとめなければなりません。これは、内容はともかく1つのゲームを完成させることに他ならず、商品を作るのと同じく大変なことなのです。商品にするには、まず、これに色をつけなければいけないでしょう。色とは、もちろんカラーのことではなく、デコレーション・ケーキのように飾りを付けるということです。具体的にはタイトルとか、デモ画面とか、画面パターンの変化などをつけることで、これらは後からいくらでも追加できます。そういう意味で、この最後のテスト・プログラムはゲームの骨組みに当たるものであり、サブルーチンの内容だけに惑わされず、データの初期設定の方法とか、その順序とか、ゲーム全体の流れを的確に把握することが、ここでの大切なポイントです。

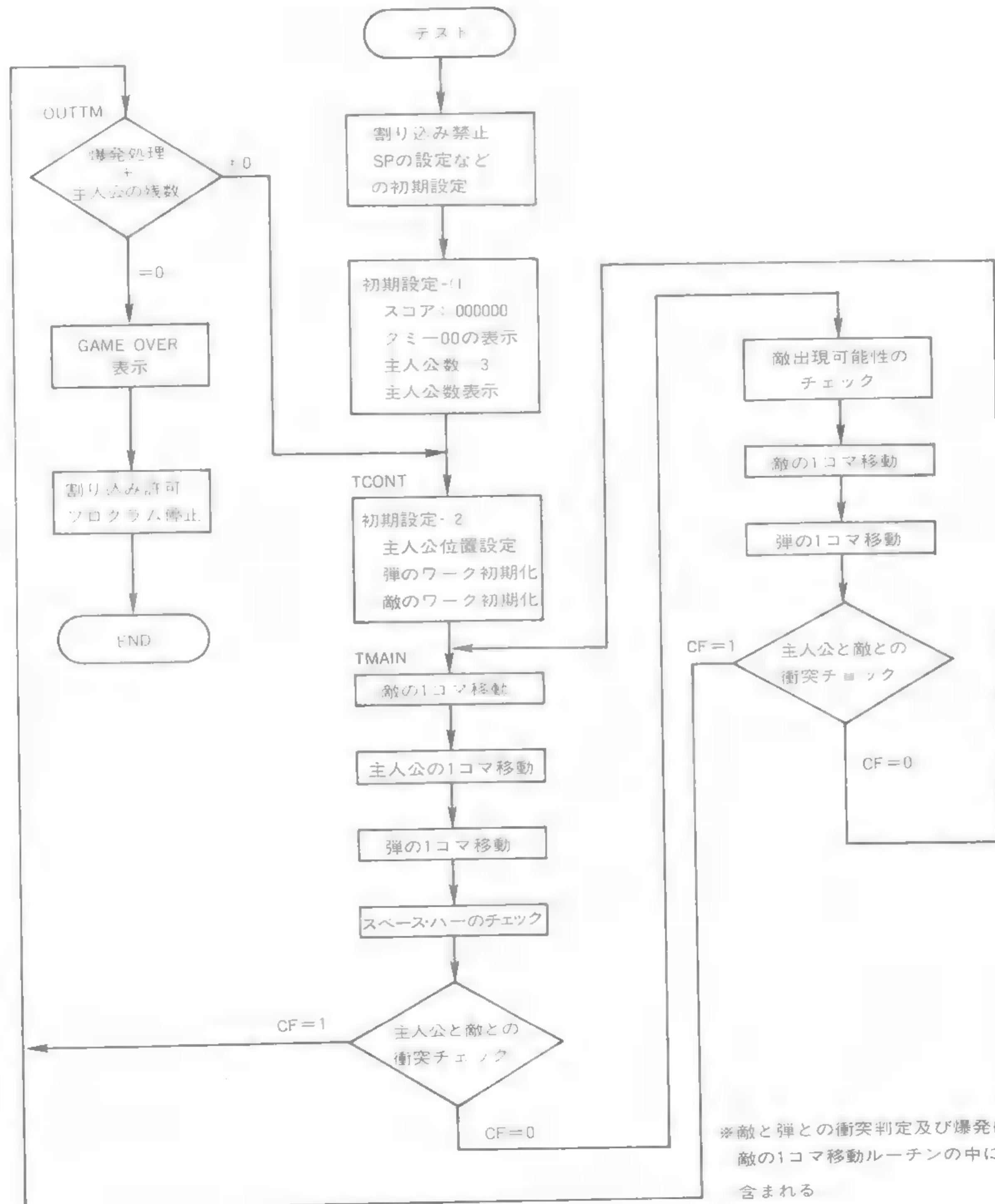
プログラムを見る前に、まずは全体をどのように構成するのか、フローチャートを見ながらその流れを追いかけることにしましょう。本来、プログラムというものはフローチャートを書いてから組んでいくと、バグの少ないものができるのですが、めんどろなためどうしても直接プログラミングしてしまうことが多くなります。複雑なプログラムは、後で見ると作った本人でも何をやっているのかわからなくなってしまうものです。大作を作る時には、できるだけフローチャートを残すことを習慣とすることを勧めます。

さて、図3のフローチャートから、敵と弾が1ループにつき2回移動するのに対し、主人公は1回しか移動していないことがわかります。これは主人公の移動速度が、敵や弾の半分のスピードであるということの意味します。また、主人公と敵との衝突判定は1ループについて2度行なわれていますが、厳密にはこれは100%の判定がされているとはいえません。それは、敵が動いて主人公と衝突の状態になった直後に、主人公が移動して敵と離れるというケースがあるからです。これを避けるには、主人公が移動する前に、もう1度衝突の判定をする必要があります。ただし、その程度のことは大目に見ようということで、今回はそこまでのきびしさは追及せず。このフローチャート通りにプログラムを組んであります。このような一見すると気がつかないような細かいことでも、フローチャートを追うことにより簡単にわかることが、めんどろなフローチャート作成の裏側にあるスバラシサの1つなのです。

プログラム本体については、まず敵の動きの中でこれまで不要部分としてコメント扱いになっていた命令を、ここで復活させています。これにより、敵の移動ルーチンの中で弾との衝突チェック、スコアのアップ、爆発時の処理(敵が弾に当たった場合)が加えられることになります。そして、この中でまだプログラミングされていない敵の爆発ルーチンと、次に出現する敵を出すルーチンが、ここで新たに組まれています。

図 3

ゲーム全体のフローチャート



プログラムの内容そのものについては、コメントを読んでいけば理解できると思いますが、新しい敵の出現位置や移動コース(これまでと違い4コースある)の選択の際に、RND(31220~31350 行)というルーチンを何度もコールしているのが目につきます。

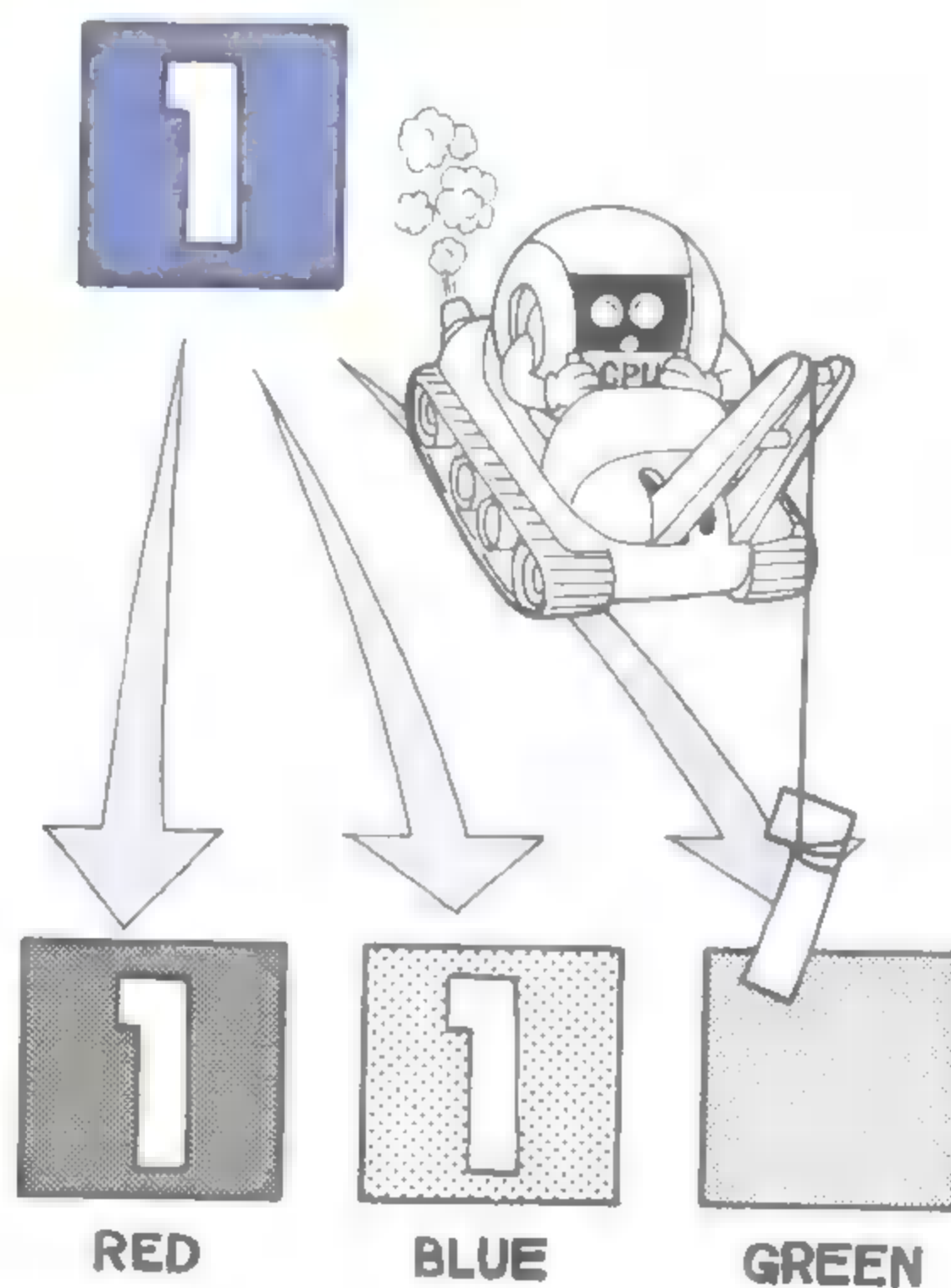
このRNDの内容については、当然乱数を発生させるものなのですが、マシン語ではBASICのように乱数を自動的に発生してくれるものなど用意されてはいません。そのため、一定の計算式によりランダムに近い数字を求める、いわば擬似乱数発生ルーチンを自分で作っておく必要があるのです。計算で求められる数字は、本来乱数とはいえないのですが、ゲームでは相手が人間ですし、ましてやリアルタイム・ゲームの中に使われた場合はその予測はまったく不可能になりますから、簡単な計算でも十分に乱数として通用するのです。ここでの乱数発生計算は、次のように行っています。

1. 乱数ワーク・エリアにある2バイトの数 $\times 5 + 3573H$
2. 1の値を乱数ワーク・エリアにストア
3. 1の値の上位バイトを乱数とする

これで、毎回違った数字が一応得られることになりますが、しょせんは同じ計算式ですからかならずループになります。ただ、それをゲーム中に暗算で出せる人など、この世にいませんから大丈夫です。もし、いたとすればその人はプログラムを見ただけで、ゲーム画面を頭に描いてプレイできるような人です。そうすると、もはや人間ではなくコンピュータそのものですから、正

真正銘の人間コンピュータといえます。なお、ワーク・エリアの初期数値や5倍した後に加える3573Hには特別の意味はなく、単なるデタラメな数です。

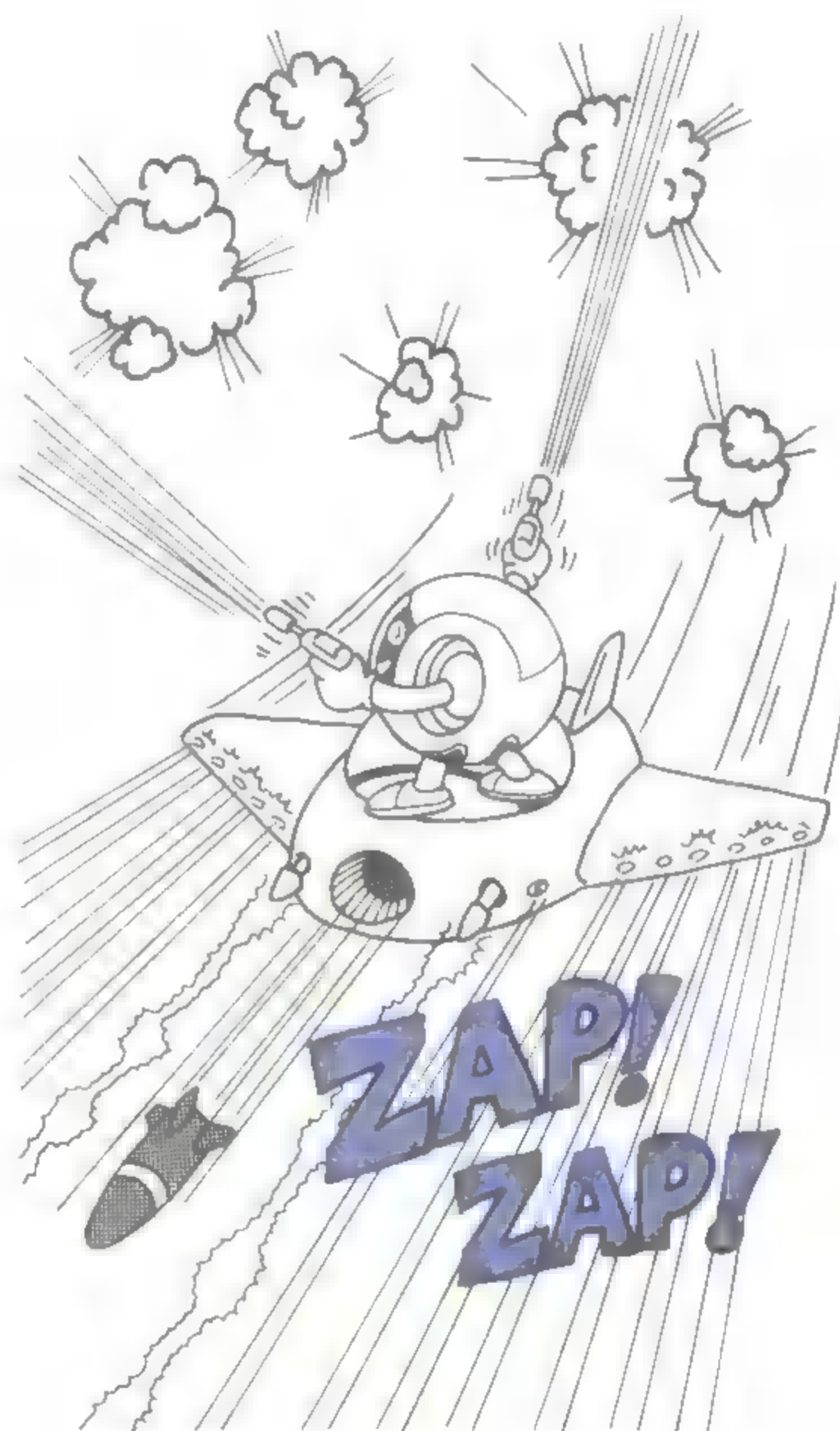
乱数ルーチンが理解できたところで、その利用方法もついでにマスターしておきましょう。出てくる数字は00-FFHですから、それを自分の作りたい数字にアレンジする必要があります。例えば、0-7までの数字が欲しい時に、最悪の方法はその数が出てくるまで何度もこのルーチンをコールすることです。もちろん、いつかは出てきますが、このような時には例のAND命令を利用



し、AND 7 とすれば一発ですべての数値は 0-7 になってしまいます。ただし、すべてがこのように一度では出てきませんから、このプログラムのように、ある程度近い数字にまで操作して、ダメならばもう一度乱数を求めるということになります。もう 1 つの方法としては、求めた乱数が 80H 以上ならば 1, 80H 未満ならば 0 というように、CP 命令を使って分けるという方法があります。この方法だと CP 命令の値を変えることにより、1 の出る確率を多くするというような乱数自体への特徴付けが可能になります。どちらの方法を取るかは、求める乱数の値と、最終的には利用するあなたの気分次第というところでしょう。

ところで、このプログラムにはもう 1 つ重要な点が隠されています。それは、始めてウェイトを取り入れたかということです。といっても、主人公が爆発する時にパターン変化の間に無駄命令があるだけのことなのですが、重要なのはこの概念なのです。ここにあるのは、とてもウェイトといえる程のルーチンではありませんが、これから本格的なプログラムに入っていくには、ウェイトをどのように入れていくかが 1 つのキーポイントになります。つまり、画面に出ている敵の数がいくつであっても、全体の速度は一定にしなければならないのです。これを無視すると、非常にカッコの悪いゲームになってしまいます。ウェイトに関しては、いずれ詳しく出てきますので、ここではその必要性があるということだけでも理解をして、このゲームをプレイしてみてください。

ゲームの内容は、次々と出てくる敵をか



わしながら弾を発射していくというだけの、単純なものです。弾の最大数は BULVAL (24360 行) で 6 になっていますから、それ以上は出ません。これを 20 にして、BULWOK (25270 行) の値も 60 にすると、弾はほぼ撃ち放題になります。こうすると、弾数の制限と同時に、速度を一定に保つウェイトの必要性が実感として感じられると思います。

2 章から続いてきたプログラムも、このゲームに関しては一応の完成ということになりました。次の章からは、また新たな部

門へのチャレンジが始まります。人生は常にチャレンジです。チャレンジする気持ちがなくなった時に、その人の青春は年齢に関係なく終わったといえます。生きるために生きる、それはもはや老後の人生でしかありません。そのような方は、どうか静かに余生をお送りください。もちろん、チャレンジ精神とはマシン語を覚えることだけではありません。時には、その若さに任せた激しいエネルギーの発散を、外に向けて

することが大切です。それには、一人で外国を旅するのが最高です。「一気イッキで飲みまくる」は、チャレンジではありません。あれは、身のほど知らずの無謀というのです。はて??? この本は、一体何の本だったのでしたっけ……。筆者自身も、段々何かが何だかわからなくなってきました。気分を一新するためにも、また次に作るゲームの参考のためにも、一遊びといきましょう。

List 3-5 シューティング・ゲームの仕上げ

```

1661 C0FA 3C          INC  A
1070 C0FB CA91C3     JP    Z,EMDEAD
1310 C12B CD71C2     CALL  EMCHK
      C12E D0        RET    NC
      C12F 2B        DEC    HL
      C130 3600      LD     (HL),0
      C132 0D3600FF  LD     (IX+0),0FFH
      C136 0D360104  LD     (IX+1),EXPL01
      C13A 0D5E06     LD     E,(IX+6)
      C13D 0D5607     LD     D,(IX+7)
      C140 CD0BC3     CALL  DISPSC

```

***** List 3-5-N *****

```

      C343          DELAY:  ;DELAY
      C343 C5        PUSH  BC
      C344          DELAYT: ;DELAY Times
      C344 0600      LD     B,0
      C346          DELAYL: ;DELAY Loop
      C346 10FE      DJNZ  DELAYL
      C348 3D        DEC    A
      C349 20F9      JR     NZ,DELAYT
      C34B C1        POP    BC
      C34C C9        RET

```

```

      C34D          ;MYCRAS: ;MY CRASH
      C34D 0610      LD     B,16
      C34F          MYCRA1: ;MY CRASH 1
      C34F C5        PUSH  BC
      C350 ED4836C2  LD     BC,(MYLOC)

```



```

C354 3E04      LD  A,EXPL01
C356 C5        PUSH BC
C357 CD00BE     CALL DISP
C35A C1        POP  BC
C35B 3E28      LD  A,40
C35D CD43C3     CALL DELAY
C360 3E05      LD  A,EXPL02
C362 CD00BE     CALL DISP
C365 3E28      LD  A,40
C367 CD43C3     CALL DELAY
C36A C1        POP  BC
C36B 10E2      DJNZ MYCRA1
C36D CDCDBE     CALL CLS
C370 110000     LD  DE,0
C373 CD0BC3     CALL DISPSC
;
C376 014A00     LD  BC,004AH
C379 2140C3     LD  HL,DUMMY
C37C CDA0C2     CALL MSGPRN
C37F 218FD0     LD  HL,MYREST
C382 35        DEC  (HL)
C383 7E        LD  A,(HL)
C384 F5        PUSH AF
C385 014A10     LD  BC,MRLOC
C388 CD92BE     CALL DISPLE
C38B AF        XOR  A
C38C CD43C3     CALL DELAY
C38F F1        POP  AF
C390 C9        RET
;
C391            ;EMDEAD: ;EnemY DEAD
C391 DD7E01     LD  A,(IX+1)
C394 DD3401     INC  (IX+1)
C397 DD4E02     LD  C,(IX+2)
C39A DD4603     LD  B,(IX+3)
C39D FE06      CP   EXPL02+1
C39F C200BE     JP   NZ,DISP
C3A2 DD360000   LD  (IX+0),0
C3A6 211004     LD  HL,410H
C3A9 CD39BE     CALL CLPTXY
C3AC C9        RET
;
C3AD            ;EMAPP: ;EnemY APPeare
C3AD 216EC1     LD  HL,EMWORK
C3B0 0603      LD  B,EMVAL
C3B2            ;EMAPPL: ;EnemY APPeare Loop
C3B2 E5        PUSH HL
C3B3 C5        PUSH BC
C3B4 7E        LD  A,(HL)
C3B5 B7        OR   A
C3B6 CCC2C3     CALL Z,NEWEM
C3B9 C1        POP  BC
C3BA E1        POP  HL
C3BB 111000     LD  DE,16
C3BE 19        ADD  HL,DE

```

```

C38F 10F1      DJNZ EMAPPL
C3C1 C9        RET

;
NEWEM: ;NEW EneMy
C3C2          LD (HL),1
C3C2 3601      CALL RND
C3C4 CD04C4    AND 3
C3C7 E603      JR Z,NEWEM
C3C9 28F7      INC HL
C3CB 23        LD (HL),A
C3CC 77        LD BC,6
C3CD 010600    ADD HL,BC
C3D0 09        LD (HL),0
C3D1 3600      DEC HL
C3D3 2B        LD (HL),A
C3D4 77        DEC HL
C3D5 2B        CALL RND
C3D6 CD04C4    AND 3
C3D9 E603      ADD A,A
C3DB 87        LD C,A
C3DC 4F        LD B,0
C3DD 0600      EX DE,HL
C3DF EB        LD HL,COUADR
C3E0 215DD1    ADD HL,BC
C3E3 09        LD C,(HL)
C3E4 4E        INC HL
C3E5 23        LD B,(HL)
C3E6 46        EX DE,HL
C3E7 EB        LD (HL),B
C3E8 70        DEC HL
C3E9 2B        LD (HL),C
C3EA 71        DEC HL
C3EB 2B        NEWEMY: ;NEW EneMy
C3EC          CALL RND
C3EC CD04C4    AND 7FH
C3EF E67F      INC A
C3F1 3C        CP DEND-20
C3F2 FE1A      JR NC,NEWEMY
C3F4 30F6      LD (HL),A
C3F6 77        DEC HL
C3F7 2B        NEWEMX: ;NEW EneMy x
C3F8          CALL RND
C3F8 CD04C4    AND 7FH
C3FB E67F      INC A
C3FD 3C        CP REND-1
C3FE FE20      JR NC,NEWEMX
C400 30F6      LD (HL),A
C402 77        RET
C403 C9

;
RND: ;RaNDom figure
C404          PUSH HL
C404 E5        LD HL,(RNDWOK)
C405 2A17C4    LD D,H
C408 54        LD E,L
C409 5D        ADD HL,HL
C40A 29        ADD HL,HL
C40B 29        ADD HL,HL
C40C 19        ADD HL,DE

```



```

C400 117335      LD  DE,3573H
C410 19          ADD HL,DE
C411 2217C4      LD  (RNDWOK),HL
C414 7C          LD  A,H
C415 E1          POP HL
C416 C9          RET

;
RNDWOK: ;RaNdOm figure WOrk area
C417 711F      DB  113,31
;
;***** List 3-5-T *****
;
D000          TEST: ;TEST
D000 F3          DI
D001 3100B6      LD  SP,STACK
D004 AF          XOR  A
D005 0351        OUT  (51H),A
D007 CDCDBE      CALL CLS
D00A 213DC3      LD  HL,SCORE2
D00D 0603        LD  B,3
D00F          TL1: ;Test Loop 1
D00F 3600        LD  (HL),0
D011 23          INC  HL
D012 10FB        DJNZ TL1
D014 014A00      LD  BC,004AH
D017 2140C3      LD  HL,DUMMY
D01A CDA0C2      CALL MSGPRN
D01D 110000      LD  DE,0
D020 CD0BC3      CALL DISPSC
;
D023 3E03        LD  A,3
D025 328FD0      LD  (MYREST),A
D028 014A10      LD  BC,MRLOC
D02B CD92BE      CALL DISPLE
D02E          TCONT: ;Test INI
D02E 2A8DD0      LD  HL,(INITML)
D031 2236C2      LD  (MYLOC),HL
;
D034 2138C2      LD  HL,BULWOK
D037 0606        LD  B,BULVAL
D039          TL2: ;Test Loop 2
D039 3600        LD  (HL),0
D03B 23          INC  HL
D03C 23          INC  HL
D03D 23          INC  HL
D03E 10F9        DJNZ TL2
;
D040 216EC1      LD  HL,EMWORK
D043 111000      LD  DE,16
D046 0603        LD  B,EMVAL
D048          TL3: ;Test Loop 3
D048 3600        LD  (HL),0
D04A 19          ADD  HL,DE
D04B 10FB        DJNZ TL3
;

```

```

50450 D040 TMAIN: ;Test MAIN loop
D040 CD58C1 CALL EMMVAL
D050 CD88C1 CALL MYMOVE
D053 CD03C2 CALL ABMOVE
D056 CDCDC1 CALL SSKCK
D059 CD4AC2 CALL MYCHK
D05C 380E JR C,OUTTM
D05E CDADC3 CALL EMAPP
D061 CD58C1 CALL EMMVAL
D064 CD03C2 CALL ABMOVE
D067 CD4AC2 CALL MYCHK
D06A 30E1 JR NC,TMAIN
D06C OUTTM: ;OUT of Test Main loop
D06C CD4DC3 CALL MYCRAS
D06F 20BD JR NZ,TCONT
D071 217CD0 LD HL,GOVER
D074 011010 LD BC,1010H
D077 CDA0C2 CALL MSGPRN
D07A FB EI
D07B FF RST 38H

D07C ;GOVER: ;Game OVER
067C D07C 47204120 DB 'G A M E'
D080 4D204520
D084 20
50680 D085 4F205620 DB 'O V E R',0
D089 45205200

50690 D08D INITML: ;INITial My Location — 主人公の初期出現座標
D08D 1A2E DB 26,46
D08F MYREST: ;MY REST — 主人公の残数が入るワークエリア
D08F DS 1

;
0004 EXPLO1:EQU 4 ;EXPL0sion 1 — 爆発パターン 1 のパターン番号
0005 EXPLO2:EQU 5 ;EXPL0sion 2 — 爆発パターン 2 のパターン番号
104A MRLOC: EQU 104AH ;My Rest LOcation — 主人公の残数表示座標
;
0001 RR: EQU 1
0002 UR: EQU 2
0003 UU: EQU 3
0004 UL: EQU 4
0005 LL: EQU 5
0006 DL: EQU 6
0007 DD: EQU 7
0008 DR: EQU 8
0009 NM: EQU 9 ;No Move — 移動しない
000A NP: EQU 10 ;New Pointer — 次の 2 バイトを方向データポインターとする
;
D090 COURSE1: ;COURSE 1 — 移動方向データ 1
D090 07070808 DB DD,DD,DR,DR
D094 08010101 DB DR,RR,RR,RR
D098 01020707 DB RR,UR,DD,DD
D09C 06060605 DB DL,DL,DL,LL
D0A0 05050504 DB LL,LL,LL,UL
D0A4 0A DB NP
D0A5 90D0 DW COURSE1
D0A7 COURSE2: ;COURSE 2 — 移動方向データ 2
50980 D0A7 07070707 DB DD,DD,DD,DD

```


D0AB 06050505	DB DL,LL,LL,LL
D0AF 05090909	DB LL,NM,NM,NM
D0B3 090906	DB NM,NM,DL;DD
D0B6 07070708	DB DD,DD,DD,DR
D0BA 01010101	DB RR,RR,RR,RR
D0BE 09090909	DB NM,NM,NM,NM
D0C2 0908	DB NM,DR
D0C4 0A	DB NP
D0C5 A7D0	DW COURS2
D0C7	COURS3: ;COURSE 3
D0C7 01070507	DB RR,DD,LL,DD
D0CB 01010705	DB RR,RR,DD,LL
D0CF 05070101	DB LL,DD,RR,RR
D0D3 01010705	DB RR,RR,DD,LL
D0D7 05050507	DB LL,LL,LL,DD
D0DB 01010101	DB RR,RR,RR,RR
D0DF 01010101	DB RR,RR,RR,RR
D0E3 07050505	DB DD,LL,LL,LL
D0E7 05050505	DB LL,LL,LL,LL
D0EB 05070101	DB LL,DD,RR,RR
D0EF 01010101	DB RR,RR,RR,RR
D0F3 01010101	DB RR,RR,RR,RR
D0F7 01010101	DB RR,RR,RR,RR
D0FB 01010705	DB RR,RR,DD,LL
D0FF 05050505	DB LL,LL,LL,LL
D103 05050505	DB LL,LL,LL,LL
D107 05050505	DB LL,LL,LL,LL
D10B 05050507	DB LL,LL,LL,DD
D10F 01010101	DB RR,RR,RR,RR
D113 01010101	DB RR,RR,RR,RR
D117 01010101	DB RR,RR,RR,RR
D11B 01010101	DB RR,RR,RR,RR
D11F 01010101	DB RR,RR,RR,RR
D123 01010101	DB RR,RR,RR,RR
D127 01010101	DB RR,RR,RR,RR
D12B 01010101	DB RR,RR,RR,RR
D12F 07050505	DB DD,LL,LL,LL
D133 05050505	DB LL,LL,LL,LL
D137 05050505	DB LL,LL,LL,LL
D13B 05050505	DB LL,LL,LL,LL
D13F 05050505	DB LL,LL,LL,LL
D143 05050505	DB LL,LL,LL,LL
D147 05050505	DB LL,LL,LL,LL
D14B 05050505	DB LL,LL,LL,LL
D14F 0507	DB LL,DD
D151 0A	DB NP
D152 C7D0	DW COURS3
D154	COURS4: ;COURSE 4
D154 08080806	DB DR,DR,DR,DL
D158 0606	DB DL,DL
D15A 0A	DB NP
D15B 54D1	DW COURS4
D15D	;COUADR: ;COURSE AddrESS
D15D 90D0A7D0	DW COURS1,COURS2
D161 C7D054D1	DW COURS3,COURS4

——移動方向データ 3

——移動方向データ 4

——移動方向のアドレス・テーブル

●音楽演奏と効果音

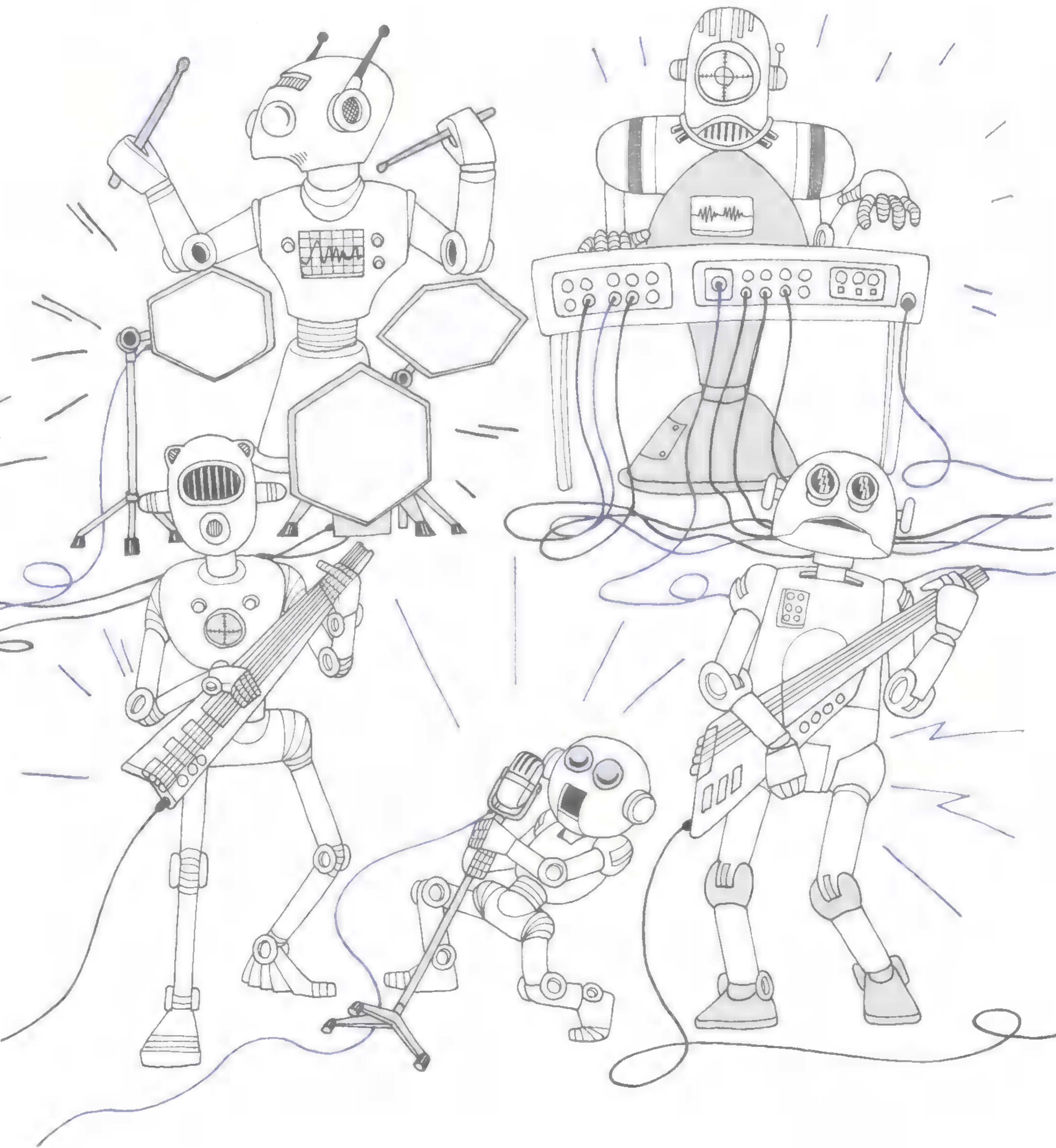
1. BEEP 音…音の仕組みとハードウェア
2. 音楽…BEEP 音楽用音程データ
3. 臨場感…BEEP による効果音
4. FM 音源とPSG…PC8801mkⅡSR 専用
5. ミュージック…FM 音源でハープシコード



●音楽は全人類共通の言葉であると言われてますが、確かにこの世から音楽が消えてしまったとしたら、寂しい世界になってしまうでしょうね。スポーツでも野球やプロレスなどかなりの分野で、音楽によって雰囲気を盛り上げて、観客をより楽しませてくれようとしています。アメリカン・フットボールなどでは、主役が一体どちらなのかわからなくなるほどハデにやっています。ゲームだって同じことです。もし、ゲーム・センターの音をすべて消してしまったら、どんな激しいゲームをしても、興味の度合いは半分以下になってしまうことでしょう。

●最新の PC8801mkⅡSR では、PSG だけでなく FM 音源というすばらしい音楽機能がついています。しかし、PC8801 や mkⅡ では、ビープ音のみで、何とも残念なことでしょう。

●そこで本章では、FM 音源、PSG を取りあげるの言うにおよばず、PC8801 や mkⅡ のビープ音を使って音楽演奏ができるようにしてみましょう。

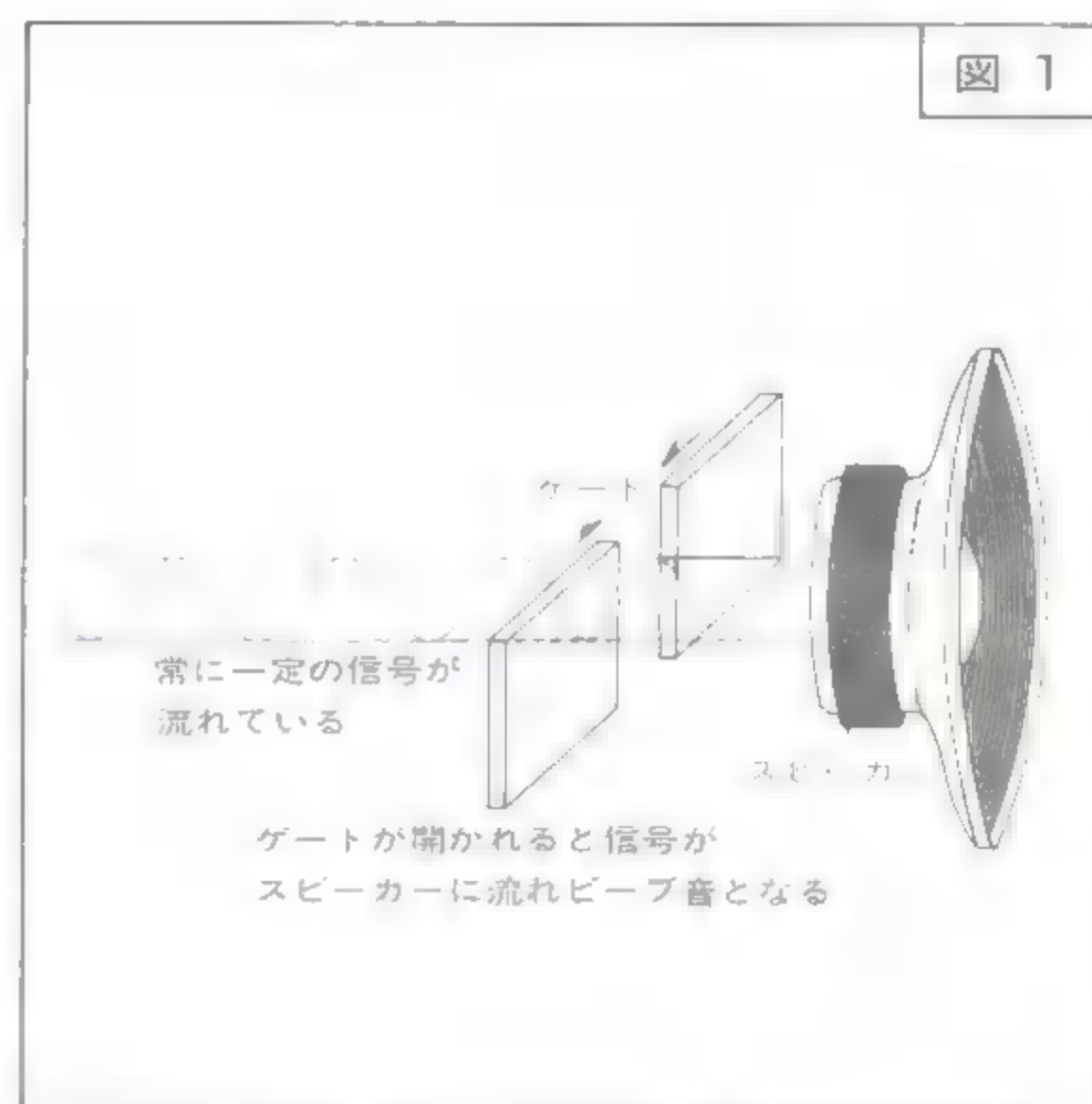


1. BEEP音…音の仕組みとハードウェア

音とは一体どういう経路で、我々の耳に聞こえてくるのでしょうか。音楽を出すプログラムを作るといっても、まず音の正体がハッキリしないことにはプログラムの組みようがありません。例えば、ドラムを叩いた時に“ドン”という音がしますが、その音の発生場所はどこなのか、また発生方法はどうなっているのか、そしてどのようにして我々の耳に音が届くのでしょうか？

簡単にいうと《空気の振動により鼓膜が震え、それを脳が音と感知している》からです。ということは、空気を震わす音源があれば音は出るということになります。タイコの場合には、空気を震わせているのがタイコの皮であり、皮に振動を与える道具がバチなのです。これに対し、ラジオやステレオなどのように音を出す電気製品の場合は、スピーカーがタイコの皮に相当し、タイコを叩くバチに当たるのが電気ということになります。

スピーカーから音を出すには、その素材であるコーンを振動させなければなりません。しかし、コーンはタイコの皮と違い、電気をオン/オフすることにより振動するのです。そして、オン/オフの間隔を狭くすれば高い音、逆に間隔をあければ低い音が出るという具合です。タイコが一定の音しか出せない理由は、この振動の周期を自由に換えられないからです。では、音の大きさはどうでしょうか。タイコは強く叩けば大きい音が出ます。スピーカーも同じように強い電気、すなわち電圧を上げれば大き



な音を出せるわけです。これを手で調節できるようにしたものが、ボリュームなのです。

PC-8801のビープ音も、電気で作られた音ですから最終的には小さなスピーカーを振動させて鳴っています。この小さなスピーカーに、電気信号を送れば音が出ることになるのですが、問題はPC-8801ではスピーカーに対する電圧の変更や電気のオン/オフは、我々がコントロールできないという点です。我々ができることは、ただ1つビープ音を出したり止めたりするだけなのです。では、なぜビープ音だけは鳴るかという、すでに一定の電圧で一定のオン/オフの周期を持った電流がスピーカーの直前まできていて、ゲートと呼ばれるものの開閉によってそれがスピーカー側に流れたり、止まったりするようになっているから

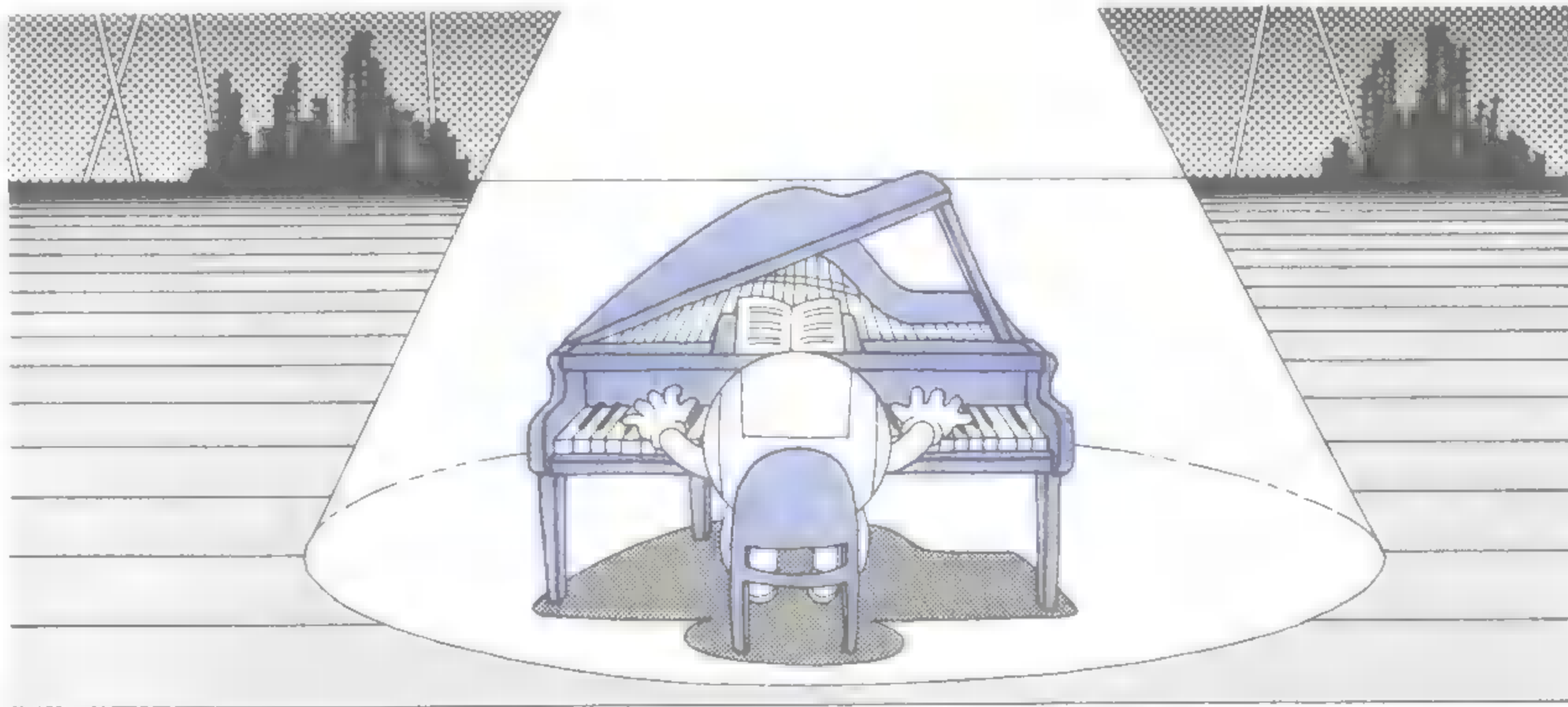
なのです。

とにかく、スピーカーを制御するにはこのゲートを操作するしか方法はないわけですから、ここを強引にオン/オフさせて音楽を作らなければなりません。その場合、当然のことながら不要なビープ音が混じってきますので、純粋に電気のオン/オフで作られた音ではなく、濁った音となってしまいます。しかし、濁った音といってもこの音しか知らなければ、濁りも全く気にならない程度のものですし、ゲームには大いに役立てることができるのです。

このビープ音の制御*をするには、出力ポート40Hのビット5を操作します。このビットを1にして、出力ポート40Hに出力すれば「BEEP 1」、0にして出力すれば「BEEP 0」ということです。これは、ビープ音だけで作る音楽ですから、ビープ音楽と呼ぶこ

とにしましょう。ビープ音楽は、音色も音量も変えられない、いわば音楽の原点です？ PC-8801mk II SRを持っている方でも、音の基礎であるビープ音楽を理解することは、FM音源やPSGをコントロールするのに大いに役に立ちますので、一通り読みながしてください。

PC-8801のハードウェア上の制約から、我々が作り出せる音は、高さと長さだけが自由で、音色はもちろんのこと音の大きさも変えられないことがわかりました。このことは、音楽的には不満が残るかもしれませんが、一方でプログラムを組むという観点から見ると、なまじ複雑なことができるよりシンプルでわかりやすいともいえます。作れる音も単音だけですから、ピアノを1本指で弾くようなものです。そこで、実際に音楽演奏のプログラムを作る前に、こ



* PC-8801mk IIでは、ボリューム変更(手動式)とスピーカーへの電流オン/オフ(出力ポート40Hのビット7を操作)がコントロールできますが、ここではPC-8801に合わせています。

のような条件下で音楽に必要な要素を具体的に考えてみることにしましょう。

音の高さ 音の長さ 休符の合図 休符の長さ

この4つの要素が確定すれば、音譜が作られることになります。ただし、ここでの楽譜がいわゆる五線譜に書くものでないことは、すでに想像がついていると思います。もちろん、最初に作曲する時は五線譜に書いて構わないのですが、コンピューターに演奏させるには、何らかの方法で16進数のデータにしなければなりません。そして、データにするということは、そのデータの開始番地とデータ終了の合図を示す必要があるということです。これらを含んだすべての要素を、実際にどのような形でプログラムに組み入れればいいのか、色々な方法が考えられますが、ここでは次のように決めています。

1. 音楽演奏の手順

- (1)HLレジスタ←音楽演奏用データの開始番地
- (2)データに基づいて音楽演奏をするルーチンをコールする

2. データの意味(終了の合図以外は2バイトで1組とする)

前半の1バイト：01-FFH = 音符または休符の長さ

00H = 演奏終了の合図

後半の1バイト：01-FFH = 音の高さ

00H = 休符の合図

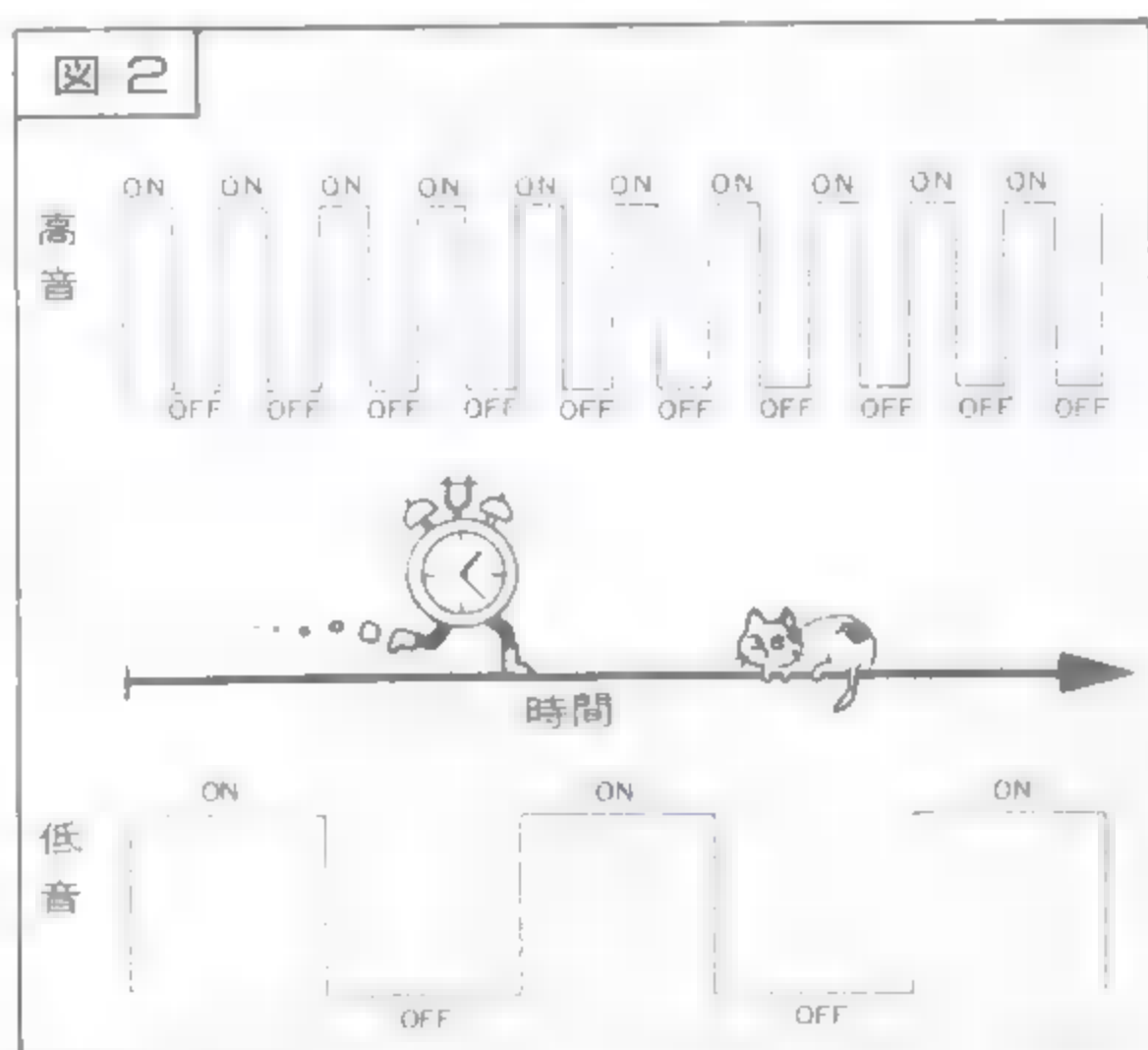
かなり具体的になりましたが、これだけの決め方ではまだ実際にプログラムを組むことはできません。それは、16進数で表わされたデータと、音との関係がハッキリしていないからです。プログラミングするに

は、この音の長さを表わす数値と、高さを表わす数値を、具体的にどのように処理するのかを決める必要があります。そこで、スピーカーに送るオン/オフ信号の実体を、図2で確認してください。

本当はこのオン/オフ信号が、純粹に電気のオン/オフであればいいのですが、残念ながらこれはビープのオン/オフを示しているのです。ですから、厳密にいうとオンの時には、ビープ音のための細かなオン/オフ信号が入っていることになるのですが、ここではそれは無視することにします。

このオン/オフ信号と音の高低との関係は、オンからオフ、またはオフからオンまでの間隔にあります。つまり、高い音ほどオン/オフの間隔が短く、逆に低い音ほどその間隔が長いということです。一方、音の長さというのは時間のことですから、オン/オフを実行しているトータル時間を計ればいいのです。ところが、実際には正確な時間を簡単に計る方法がないので、繰り返したオン/オフの回数によって、その長さを表わすことにしています。したがって、図2の例のように同じ長さの音でも、音の高さによって、長さを示す値(オン/オフの回数)は違ってくることになります。

また、音の高さ(オン/オフの間隔)の表現は、これも時間で表わします。しかし、数えるものが何もありません。そこで、何か基準となる無駄命令を決めて、その実行回数を数えるという原始的な方法で数値化します。音を微妙に変えられるようにするには、この無駄命令もできるだけ簡単なものの方がいいことになりますから、最も単純に「実行回数-1」をするだけにします。



これで、音の長さと高さを数値に変更する方法が決まりました。休符については、オン/オフの代わりにオフ/オフと実行させれば、無音状態にすることができます。データの内容が決まれば、残るはプログラムだけです。(HL)にあるデータの音楽演奏をするには、次のような流れにすればいいのです。

- 1) $B \leftarrow (HL)$ …音符, または休符の長さ
- 2) $B=0$ なら演奏終了: $HL \leftarrow HL+1$
- 3) $C \leftarrow (HL)$ …音の高さ
- 4) $C=0$ なら 9)へ
- 5) BEEP 1: ウェイト
- 6) BEEP 0: ウェイト
- 7) $B \leftarrow B-1$: $B < 0$ なら 5)へ
- 8) $HL \leftarrow HL+1$: 1)へ
- 9) 休符(ウェイト $\times 2 \times B$): 8)へ

「ウェイト」… $C=0$ になるまで $C \leftarrow C-1$ を繰り返す

これを実際にプログラミングしたのが、List 4-1 です。一部、現時点では意味のわか

らない箇所(ラベル名で POINT 1 と POINT 2)もあると思いますが、今は無視してください。なお、本章のプログラムは新たに作成するものです。2, 3 章のリストとマージする必要はありません。グラフィック関係などには、これまでのものと共通して使えるものもあるのですが、プログラム・ミスを防ぐ意味からも、あえて書き直しています。

テストの実行は、例によって D000H 番地からですが、まだ演奏用のデータが何も入っていないから、適当な数字を D500H 番地から 2 バイトずつ入れてデータとします。テストですから、10 バイトほどあれば十分です。データの最後には、演奏終了の合図 00 を入れることも忘れないでください。

h]GD000 [4]

多分、音楽にならない迷曲が演奏されたと思います。演奏が終わってもテキスト画面が消えたままなので、暴走したのでは…と、一瞬不安になった方もいるかもしれませんが、こういう時は暗闇の中で次のようにすれば良かったのですね。

h]^b
WIDTH 80 [4]

テキスト画面が見えるようになったところで、このプログラムで実際に音楽を演奏させるには、音程を表わすキチンとしたデータ表が必要です。この音程データ表というものは、本当は自分で苦労して作るべきものなのです。しかし、お急ぎの方には《トラの巻》があります。それは、すなわち次節へ進むことですが…。

List 4-1 BEEP 音の演奏

0000

```

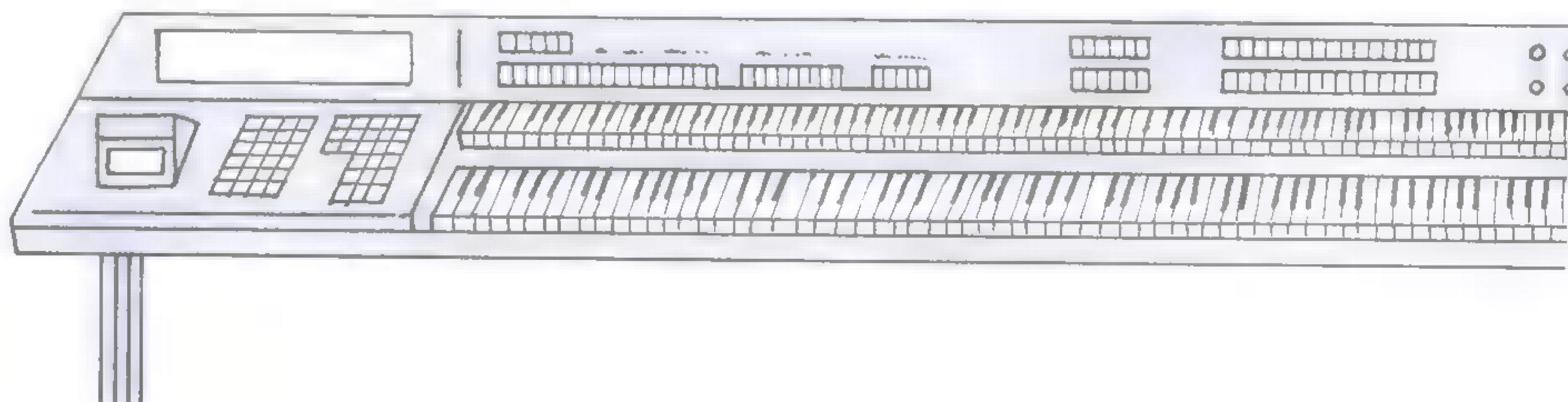
;***** List 4-1-N *****
;
;      ORG 0C000H
;
;MUSIC: ;MUSIC play
;      XOR A
;      LD B,(HL)
;      CP B
;      RET Z
;      INC HL
;      LD C,(HL)
;      CP C
;      JR Z,PAUSE
;
;BEEP1: ;BEEP 1
;      LD A,(0E6C1H)
;      OR 20H
;      OUT (40H),A
;      CALL WAIT
;POINT1: ;POINT 1
;      NOP
;      LD A,(0E6C1H)
;      AND 0DFH
;      OUT (40H),A
;      CALL WAIT
;POINT2: ;POINT 2
;      NOP
;      DJNZ BEEP1
;      JR NEXTDT
;

```

C000
C000 AF
C001 46
C002 B8
C003 C8
C004 23
C005 4E
C006 B9
C007 281A

C009
C009 3AC1E6
C00C F620
C00E D340
C010 CD2EC0
C013 00
C014 3AC1E6
C017 E6DF
C019 D340
C01B CD2EC0
C01E 00
C01E 00
C01F 10E8
C021 1808

;BEEP1: ;BEEP 1
LD A,(0E6C1H)
OR 20H
OUT (40H),A
CALL WAIT
;POINT1: ;POINT 1
NOP
LD A,(0E6C1H)
AND 0DFH
OUT (40H),A
CALL WAIT
;POINT2: ;POINT 2
NOP
DJNZ BEEP1
JR NEXTDT




```

C023      PAUSE:  ;PAUSE
C023 CD2EC0      CALL WAIT
C026 CD2EC0      CALL WAIT
C029 10F8        DJNZ PAUSE
C02B      NEXTDT: ;NEXT DaTa
C02B 23          INC HL
C02C 18D2        JR MUSIC

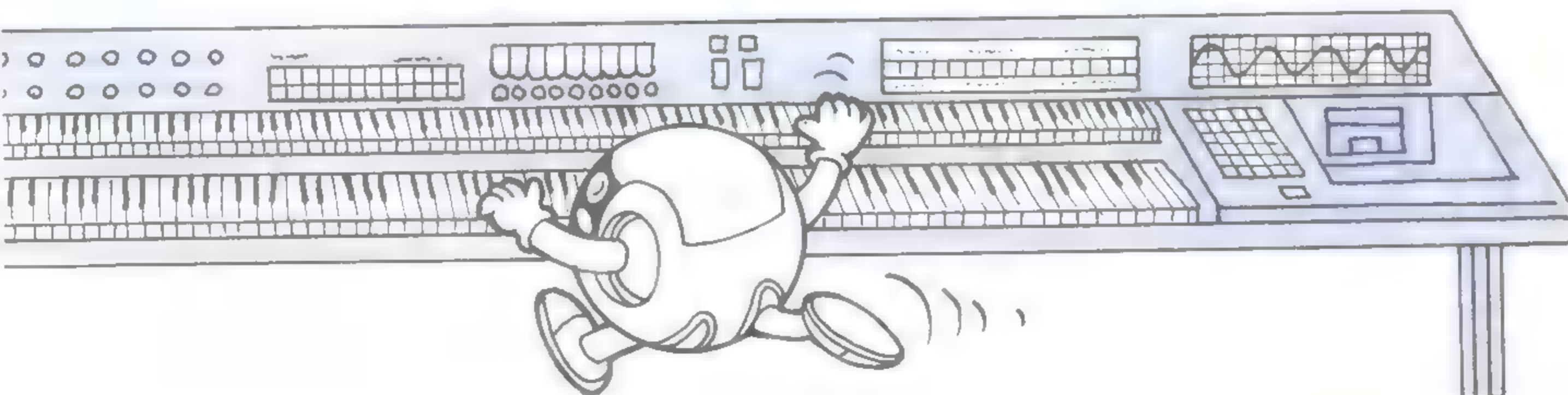
;
C02E      WAIT:  ;WAIT
C02E C5          PUSH BC
C02F      WCOUNT: ;Wait COUNTER
C02F 00          DEC C
C030 20FD        JR NZ,WCOUNT
C032 C1          POP BC
C033 C9          RET

;
***** List 4-1-T *****
;
ORG 0000H

;
TEST:  ;TEST
DI
LD SP,STACK
XOR A
OUT (51H),A
LD HL,00500H
CALL MUSIC
EI
RST 38H

;
B600      STACK: EQU 0B600H ;STACK pointer

```



2. 音楽…BEEP 音楽用音程データ

音程とは、一体どのようにして決められているのでしょうか。これが、むずかしいようで実は非常に簡単な取り決めしかしていないのです。基準となる音はハ長調のラで、周波数(1秒間のオン/オフの回数)は440HZとなっています。そして、音程が1オクターブ上がれば周波数は倍に、下がれば半分になります。1オクターブをピアノで見ると、黒鍵も含めて12のキーが並んでいますから、周波数も12に分ければいいのですが、均等にわけるのはではなく、1オクターブ上がった時に倍になるようにしなければいけません。

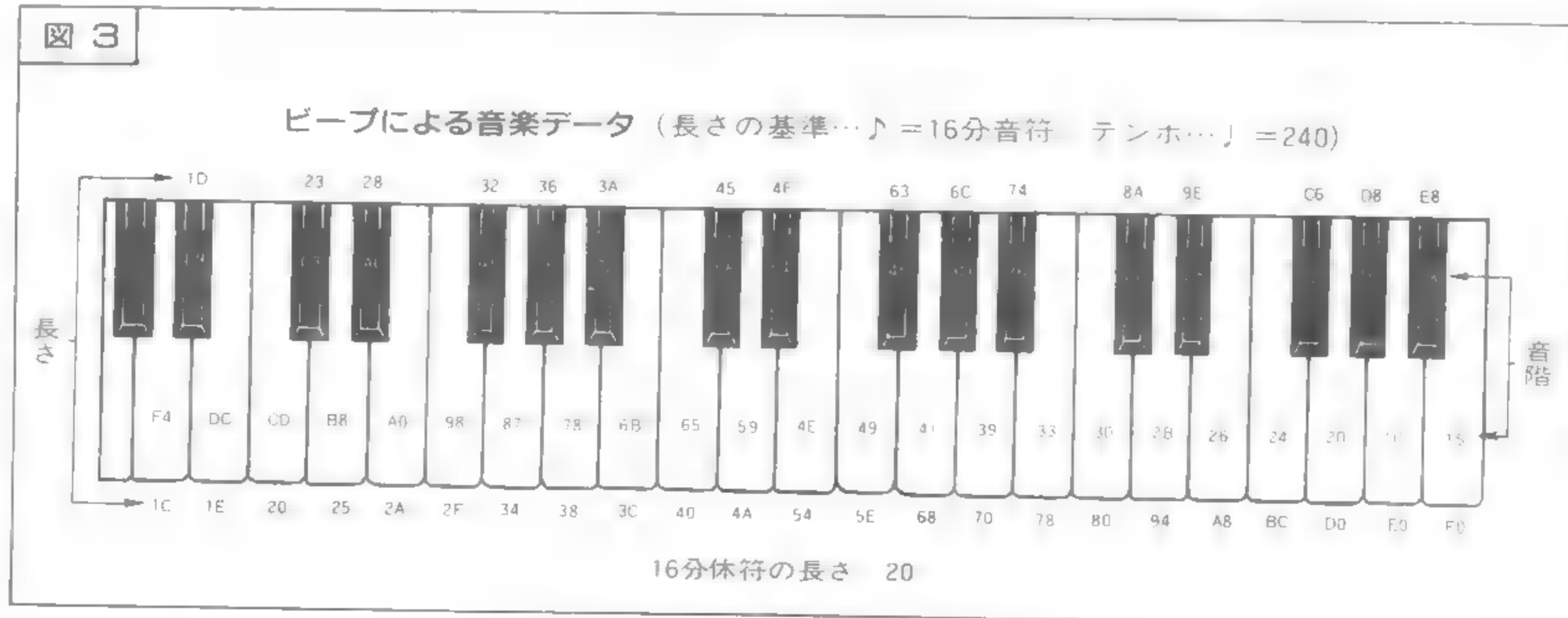
これだけの決まりなので、基準音だけわかれば音程データの作成は単なる作業になりそうですね。ところが、このビープ音楽というのは濁りはあるし、正確なメトロノームもないという、いわば問題だらけの音楽ですから、完全な音程など作りようがありません。一応、このことを頭に入れた

上で、図3の音程データ表を見るようにしてください。

このデータ表によると、音の高さが1オクターブの差で数字がキチンと倍(半分)になっている時と、大分ズレている時とありますが、これもビープ音楽に問題のある証拠で、計算通りのデータで実際にテストをしてみると、音程が狂ってしまうのです。

そのため、この音程データは試行錯誤の結果作成した貴重な資料なのです。しかし、私が音楽的にプロの耳をもっているというわけではありませんから、あるいは音程に若干の狂いがあるかもしれませんし、音の長さも正確ではないかもしれません。そのあたりのデータ修正に関しては、あなたなりの音に仕上げるようにしてください。この音程データを基にして作った、テスト音楽がありますので実験してみましょう。モニタを使って、次ページのデータをD500H番地から入力してください。

図 3



アレ!! どこかで聞いたことがある…と、すべての方が思ってくださいと非常にウレシイのですが…

この音楽のデータを見ると、音程データ表(図3)そのままの使い方ではありません。例えば、長さなどはまったくデタラメのようですし、小さな休符も意味もなく多く使われているように見えます。実は、この辺がビープ音楽の長所でもあり、まためんどうな点でもあるのです。できるだけ長所を生かすには、次のことを考えながら、少しずつ修正を加えて完成させるようにするしかありません

1. 連符や音の歯切れ良くしたい時は、間に短い休符を入れる

例: ド・ド・ド = ド・短い休符・ド・短い休符・ド

2. 音の長さは、実際に耳で聞き何度も修正をする

1. の例でも短い休符が入る分だけ音が長くなることになりまして、楽譜通

りのデータは中々できないものです。数値ですから、音符にできないような微妙な長さでも構わないのです。聞きながら、気に入るまで修正してください。

3. 譜面では表現不可能な高さの音も作ることができる。

この音楽の最初の部分でも使っているテクニックですが、音程データを少し(+1 または -1) 狂わすことにより、音をずらすことができます。また、ミとファの中間というような楽譜にない高さの音が作れるのもデジタル・ミュージックの面白さです。

これで、実際の演奏用データがキチンとまらない理由がわかったと思います。自由ということは、すなわちめんどうということなのです。ピアノよりエレクトーン、エレクトーンよりシンセサイザー……音が自由になるにつれて操作するスイッチ類が多くなっていくようなものです。

MON 

h]ED500 

```
D500 A0 41 90 40 A0 41 90 40 05 00 A0 4E 90 4F AD 4E
D510 90 4F 07 00 50 59 0A 00 50 65 0A 00 50 59 0A 00
D520 50 4E 0A 00 F0 65 0A 00 A0 87 0C 00 B0 65 0A 00
D530 58 59 0A 00 60 4E 0A 00 63 49 0A 00 68 41 0A 00
D540 6B 39 0A 00 71 33 0A 00 FF 30 FF 30 FF 30 00 .
```

.

.

h]GD000 

3. 臨場感…BEEPによる効果音

ゲームの進行を側面から盛り上げるという意味では、ビープ音楽も1つの効果音といえます。一般的には音楽でない音のことを効果音といいます。つまり、ゲーム・センターなどにあふれている例の音のことです。

ビープ音の波形(図2)を、もう1度見てください。高音でも低音でも、オンの時間とオフの時間が同じになっています。別に、わかりやすくするために同じ間隔にしているわけではありません。この間隔を一定にしているからこそ、一定の高さの音がでているのです。もし、この間隔がバラバラだったら、それはもう雑音でしかないのです。それでは、図4のようにある決まったルールの下で、この間隔を変化させたらどうなるでしょうか。

実際にどんな音になるのか、想像がつかないかもしれませんが、例1では高音から

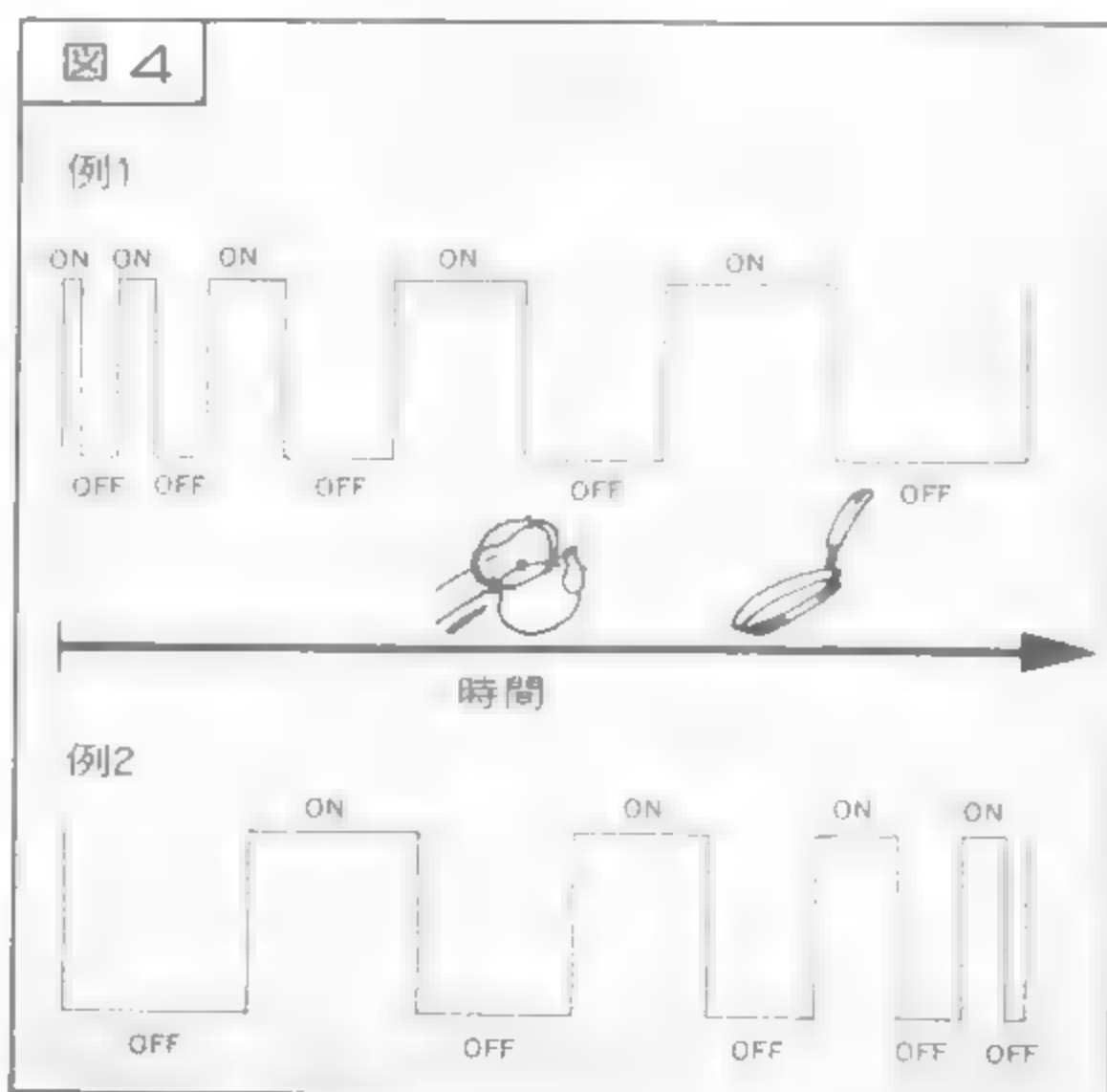
低音へ、例2では低音から高音へ、いずれも急激に変化していくハズです。この図4では、ビープ・オフの後に間隔の変更がされています。ということは、List 4-1の中で間隔を示しているのはCレジスタですから、ビープ・オフ後にCレジスタの値を+1、または、-1すればいいということになります。

このCレジスタ値の変更場所を、具体的にみるとList 4-1のPOINT 2(20250～20270行)がそれに当たります。つまり、ここを00から0CH(ニーモニックでINC C)とすれば例1のようになり、0DH(ニーモニックでDEC D)とすれば例2のようになるわけです。同じようなことをPOINT 1(20190～20240行)でも実行すれば、音の変化はより急激になることになります。結局、全体では次頁の4種類の音変化ができます。

この4種の音変化をさせるプログラムがList 4-2ですが、それぞれ実行が終わると変更したポイントを00に戻すようにしてあります。このプログラムはList 4-1とMergeしたらセーブしてください。5章では、このセーブしたプログラムにMergeして迷路ゲームを作ります。アセンブルしたら、D500_H番地から適当な数を入れて走らせてください。データ終了の合図00を入れるのも、先ほどと同じです。

h]GD000 ☐

どんな音になりましたか。インベーダーの襲来を思わせるような、そんなカッコイ



- (1) POINT 1 のところを 0DH(DEC C)音のアップ変化 1
- (2) POINT 2 のところを 0CH(INC C)音のダウン変化 1
- (3) POINT 1, POINT 2 のところを 0DH(DEC C)音のアップ変化 2
- (4) POINT 1, POINT 2 のところを 0CH(INC C)音のダウン変化 2

イ音になった方もいるかもしれません。同じデータでも効果音のコール先を4種類変えて実験してみると、色々な音に変化するはずです。データによっては、かなり面白い音が作れると思います。ただし、こう

いう特殊音にはデータ表などありませんから、すべて自分で記録管理しておかないと、イザという時に毎回テストの連続ということになってしまいます。その辺は自分なりに工夫をしてください。

List 4-2 BEEPによる効果音

```

;***** List 4-2-N *****
;
C034          SND1: ;SOUND 1          ;効果音 1
C034 3E0D      LD  A,0DH
C036 180D      JR  CPT2
C038          SND2: ;SOUND 2          ;効果音 2
C038 3E0C      LD  A,0CH
C03A 1809      JR  CPT2
C03C          SND3: ;SOUND 3          ;効果音 3
C03C 3E0D      LD  A,0DH
C03E 1802      JR  CPT12
C040          SND4: ;SOUND 4          ;効果音 4
C040 3E0C      LD  A,0CH

C042          ;
C042          CPT12: ;Change Point 1 & 2
C042 3213C0    LD  (POINT1),A
C045          CPT2: ;Change Point 2
C045 321EC0    LD  (POINT2),A
C048 CD00C0    CALL MUSIC
C04B AF        XOR  A
C04C 3213C0    LD  (POINT1),A
C04F 321EC0    LD  (POINT2),A
C052 C9        RET

;***** List 4-2-T *****
;
D000          TEST: ;TEST
D000 F3        DI
D001 3100B6    LD  SP,STACK
D004 AF        XOR  A
D005 D351      OUT  (51H),A
D007 2100D5    LD  HL,0D500H
D00A CD34C0    CALL SND1
D00D FB        EI
D00E FF        RST 38H

;
B600          STACK: EQU 0B600H ;STACK pointer

```

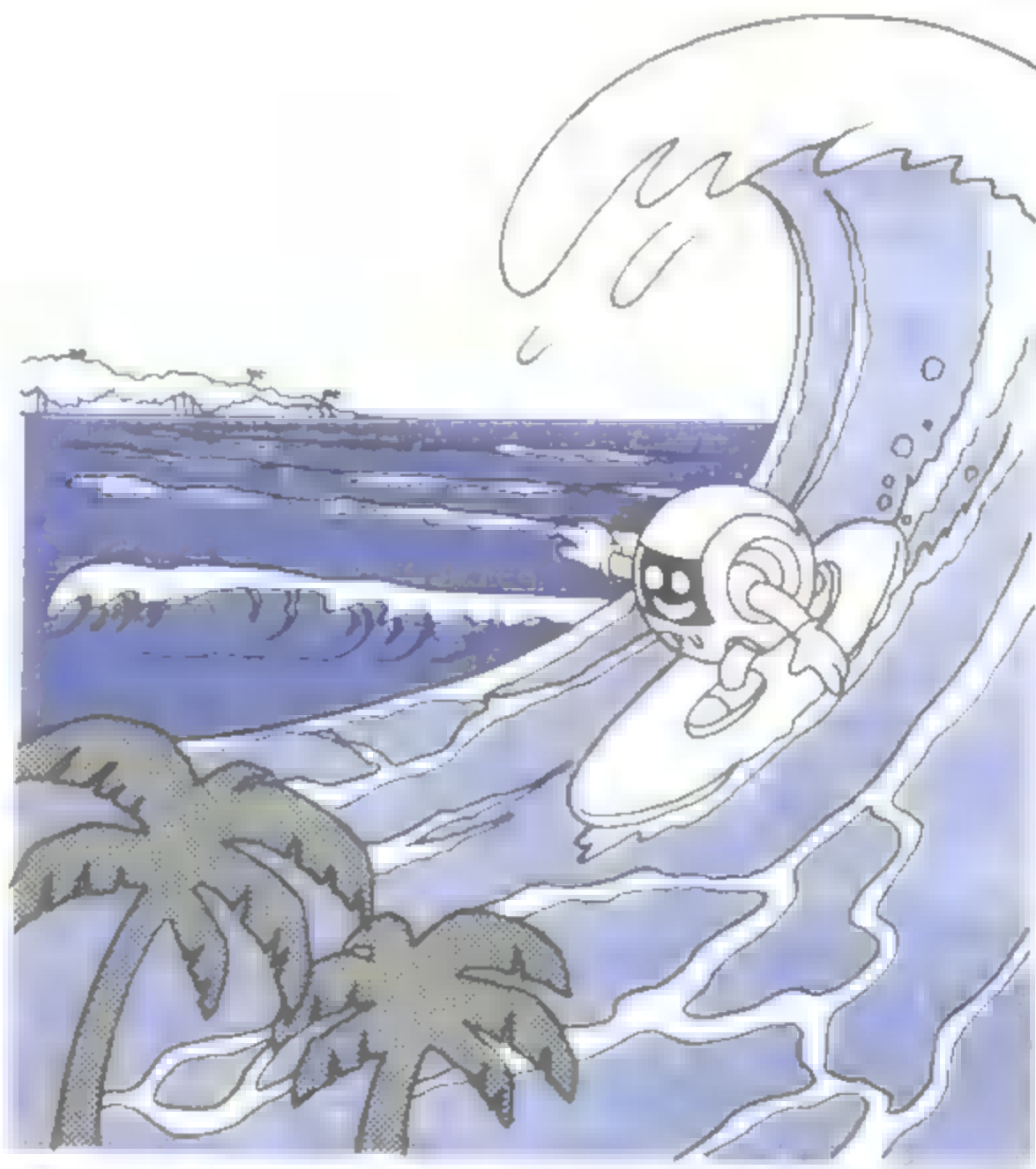
4. FM音源とPSG…PC8801mkⅡSR専用

本節と次の5節は、PC-8801mkⅡSRをお持ちでない方は、実際にテストをすることはできません。しかし、将来のために一応読んでおいても損にはならないと思いますが、そのあたりの判断はオマカセいたします。

音は空気の振動ですから、その振動波の間には、必ず変化に要する時間があるわけです。したがって、本当の音の原点とは図5にあるようなサイン波のことをいいます。

しかし、このようなキレイなサイン・カーブだけでは世の中にある様々な音を再現することはできません。世の中の音というのは様々なノイズが入って1つの音となっているわけです。そのような音を電氣的に作る出すには、この基本のサイン波に色々な細工をして、似たような音の波に加工しなければなりません。これを擬似的にできるのがPSGであり、本格的にできるのがFM音源なのです。

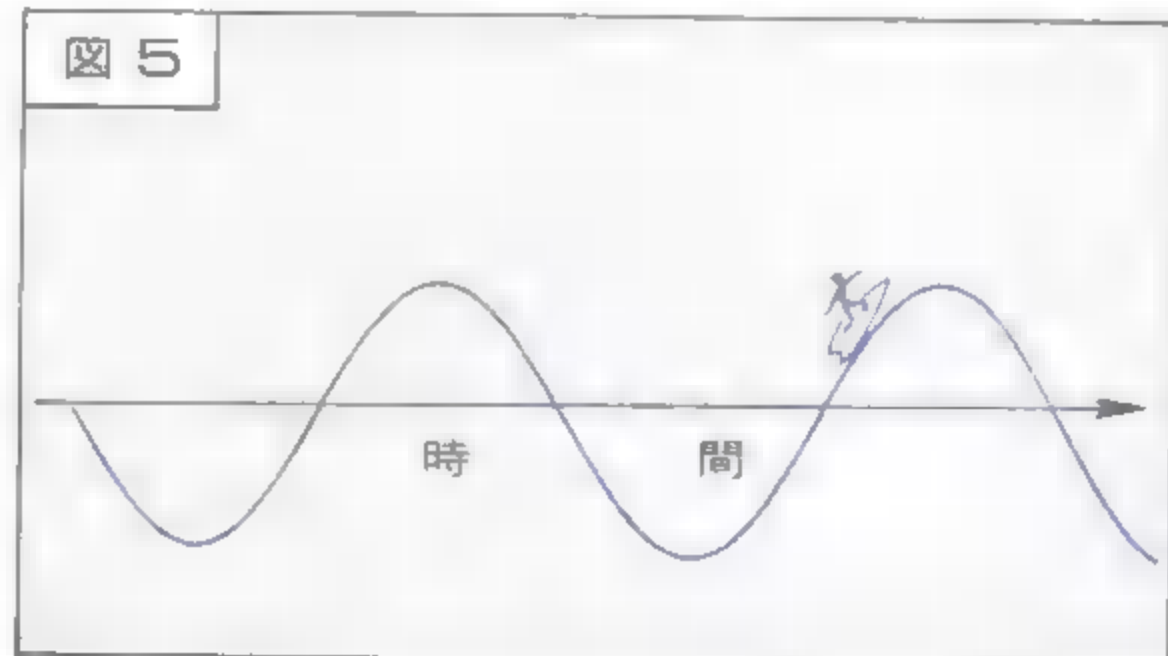
FM音源というのは、1つのシンセサイザーですからどんな波形の音でも、理論的には作り出すことができます。極端なことをいえば、あなたの好みの歌手の声を作ること

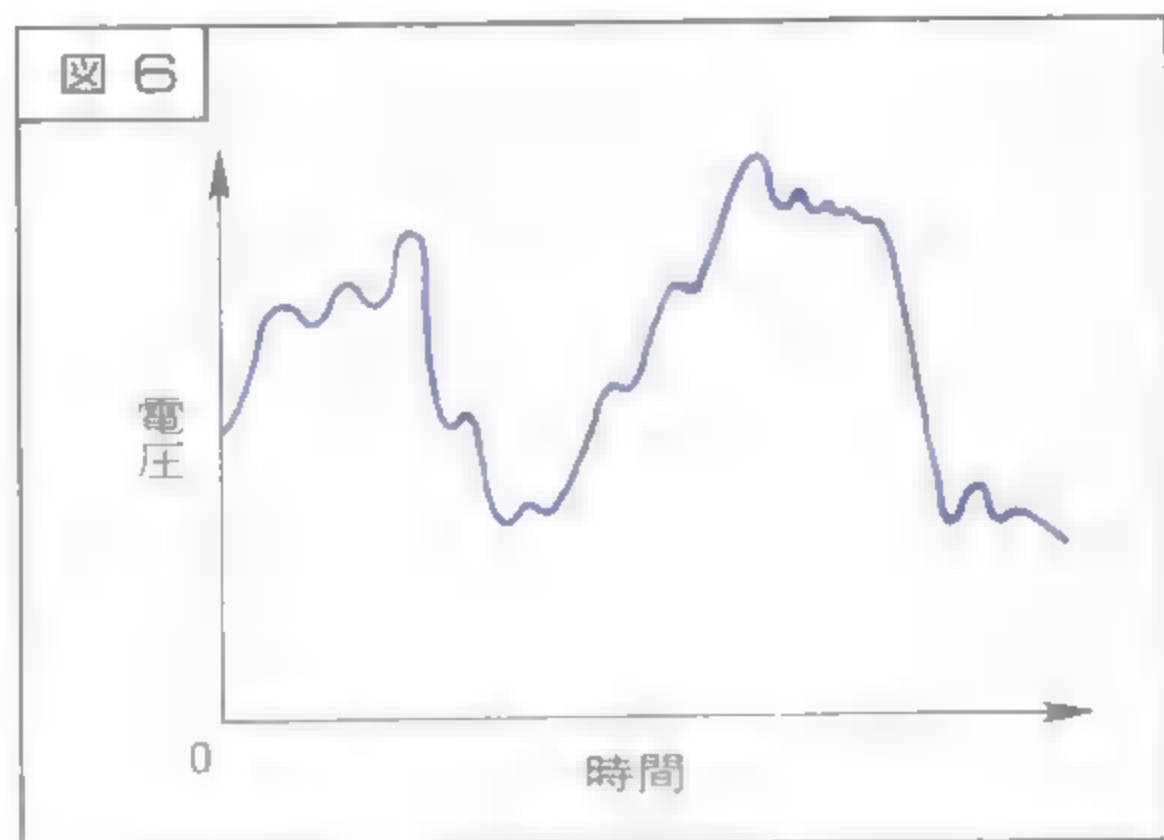


も可能なわけです。ただし、これをするためにはオシロスコープを使ってその声の波形を分析し、それと同じ波形を試行錯誤しながら作り出さなければなりません。現実的には、このようなことが誰にでもできるものではありませんし、偶然に期待するか方法はなさそうです。それでは、せっかくのFM音源がまるで宝の持ち腐れになってしまうのではないかと、ということになりますが、そのために最初からメーカー側で、色々な楽器の音や効果音を用意してくれているのです。BASICで音色番号を指定すれば、ピアノや小鳥のさえずり音が簡単に出てくるのはそのためです。

これに対し、PSGというのは3つの音源

図5





と1のノイズで構成されており、音色の決まった音楽演奏はできても、効果音の用意などはありません。そのため、効果音を出すには音源とノイズを組み合わせたり、エンベロープ(時間的な音量変化)をかけるなどして力で作るようになります。ですから、FM音源のようにどんな音でもできるというわけにはいきませんが、それでも工夫次第で色々な効果音を作ることが可能です。

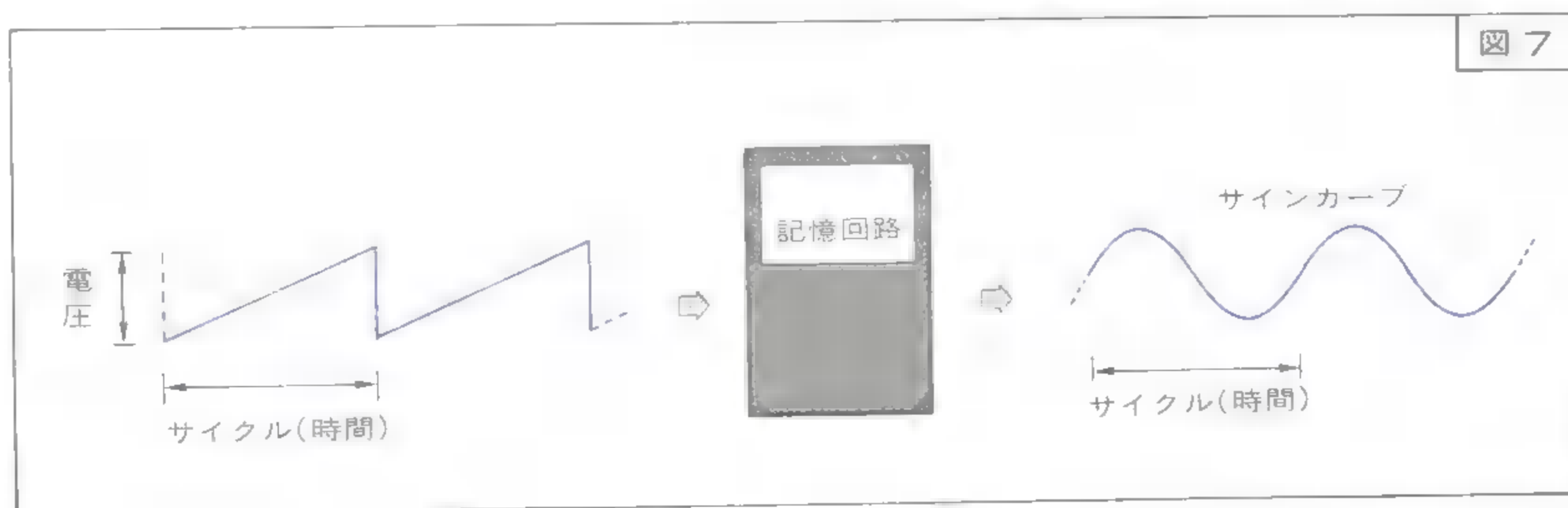
このような特徴のあるFM音源とPSGを両方共兼ね備えたPC-8801mk II SRですが、ここでその使用方法すべてを書くことはスペース的に無理です。それだけで、それこそ1冊の本になってしまいます。そこで、ここではFM音源の基本的な原理と、

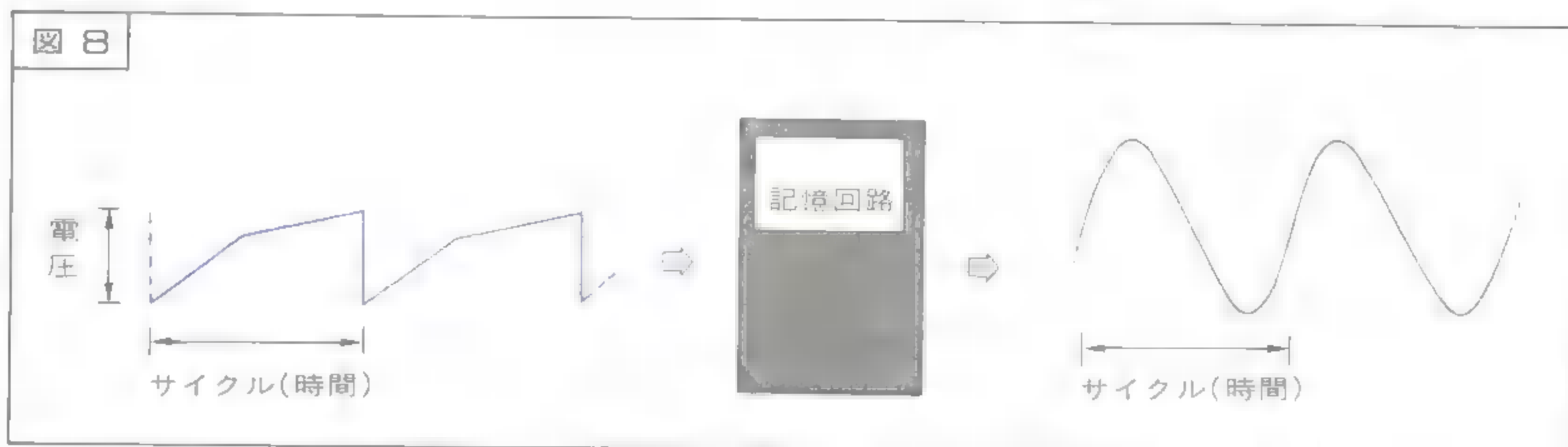
マシン語によるFM音源コントロールについて、とにかくドレミを出すということを最終目標にすることにしました。

まず、スピーカーに送る信号ですが、音色が加わることによりビープ音楽よりずっと波形が複雑になります。これは時間の変化と共に複雑な形の電圧変化が加わるということです。

この図6にある波形は単なる例であって特別な意味はありませんが、このような複雑な波形をどのようにして作り出していくのか、を説明しなければなりません。そのためには、FM音源の基礎となっているノコギリ波とサイン波の関係を理解する必要があります。というのは、FM音源ではすべての音を2つの波形の合成という形で作っており、その基本となるのがノコギリ波とサイン波であるからです。そして、どんな複雑な波形もこの合成を何度も繰り返すことによって、作成可能になるのです。このような合成の基本となるのは、簡単にいうと図7に示されるようにノコギリ波から生み出されるサイン波にあります。

この図7にある記憶回路の中には、サイ





ン波の素が入っています。このサイン波の素というのは、入ってくる電圧によって出る電圧を決定するもので、図7のように一直線的に上昇する電圧の時に、きれいなサイン・カーブを描きます。したがって、出てくるサイン波の周波数を決めているのはノコギリ波であり、ノコギリ波の周波数がそのままサイン波の周波数になるのです。また、ノコギリ波の電圧変化率(数学的にいうと傾き)を途中で変えると、出てくるサイン波の形も途中で変わってきます。例えば図8ではノコギリ波の電圧が前半は急激に、後半は穏やかに上昇しているため、出てくるサイン波は標準形とは違って前半のカーブが急になっています

このような関係から、記憶回路に入れるノコギリ波をより複雑にすれば、出てくる

波形はさらに複雑に変化することが想像できます。そのため、実際にはノコギリ波とサイン波を先に加算合成し、複雑な形にしてから記憶回路に入力します。そして、記憶回路から出力された波形にエンベロープ(掛け算合成により時間的な音量変化をつけること)をかけて、1つの波形ができ上がります。加算合成と掛け算合成の実体は、図9のようなものです

このノコギリ波とサイン波を加算したものを記憶回路に入れ、出てきた波にエンベロープをかける、ということを基本構成としたものをオペレータといいます。PC-8801mk II SRに搭載されているシンセサイザーICは、1つの音に対して4つのオペレータを持っており、図10にあるような組

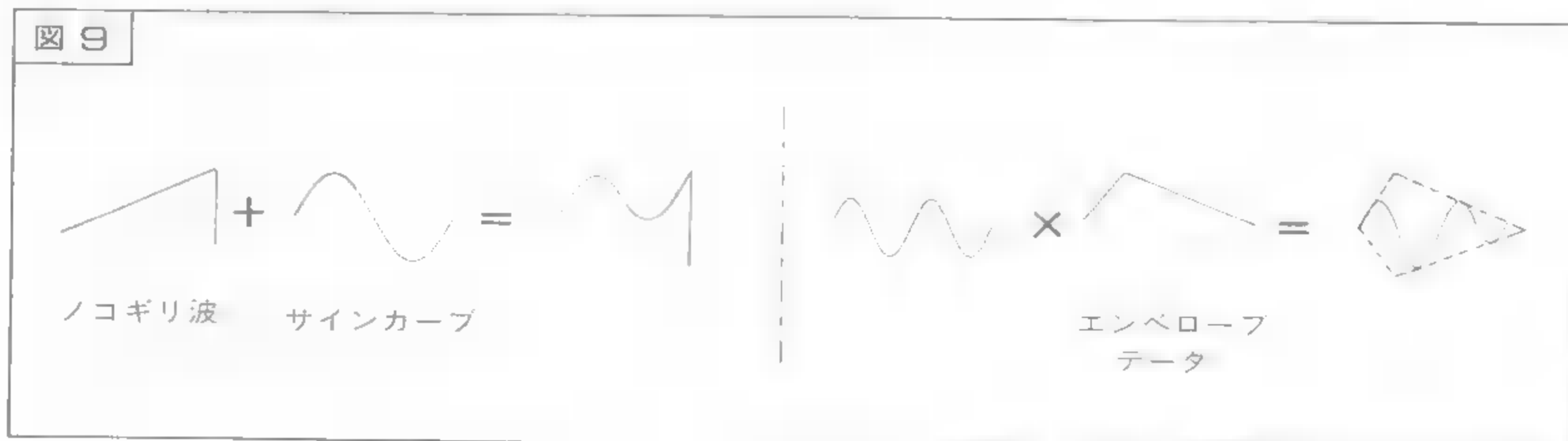
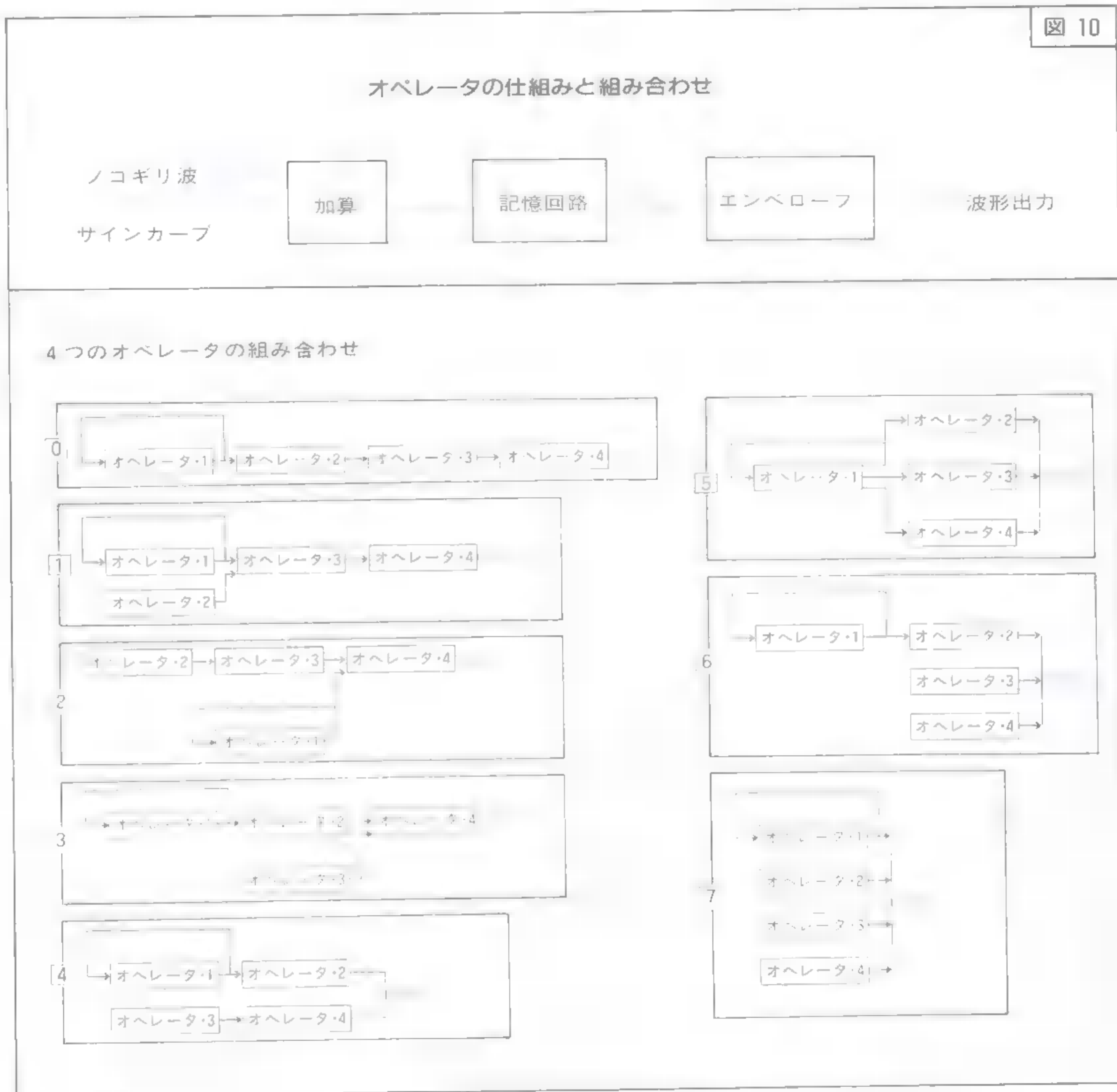


図 10



み合わせで、さらに複雑な波形を出力するようになっています。これらのオペレータの内、一番最後にある音の波形を出力しているものをキャリアといい、キャリアに対して変調用の波形を送るオペレータのことをモジュレータといいます。また、このようなオペレータの組み合わせ

(接続の仕方)のことをアルゴリズムといいます。

以上がFM音源の最も基本的な概念ですが、実際に自由な音作りをするにはこれらの知識に加えて、そのコントロール方法を理解しなければなりません。本書では、FM

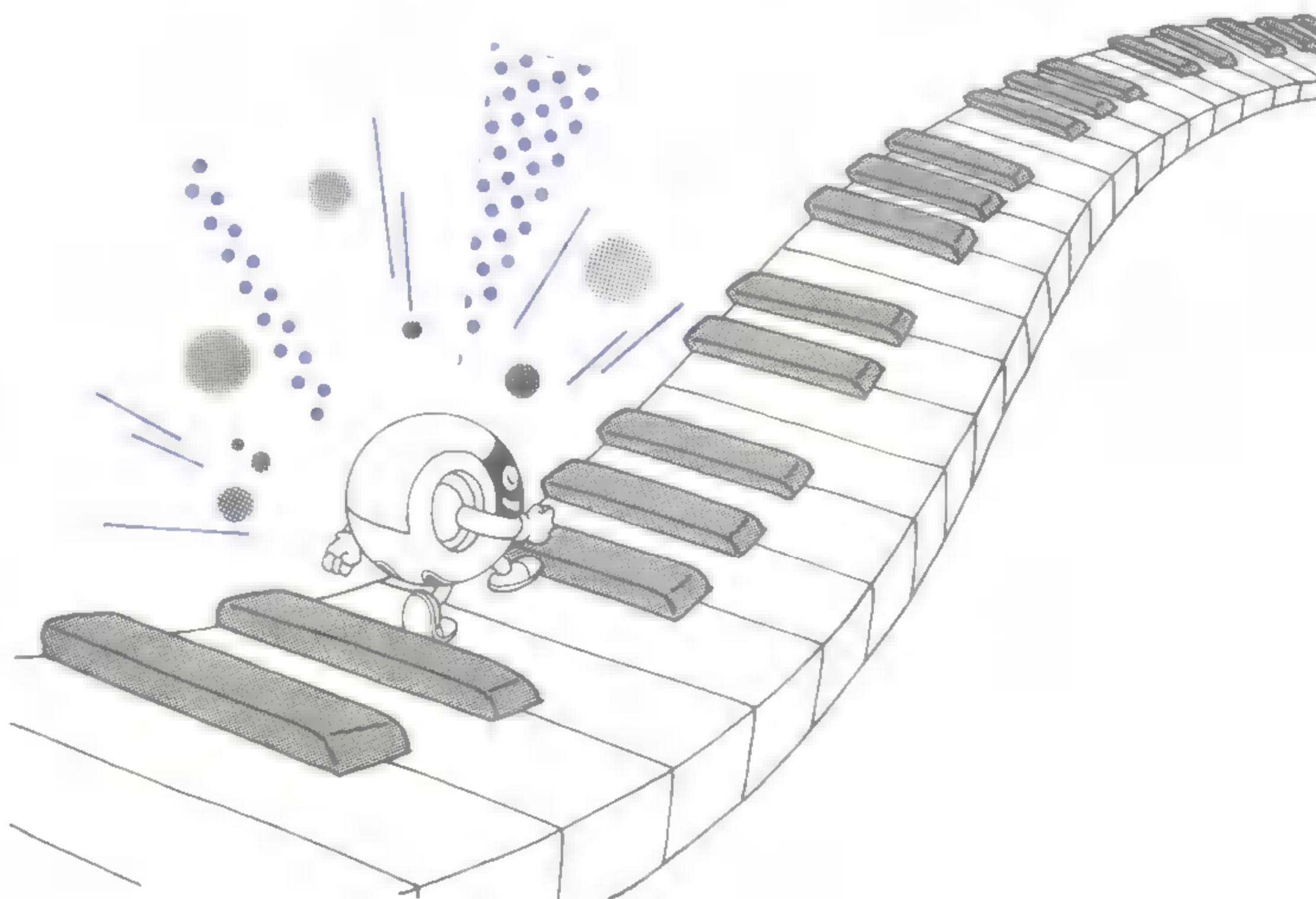
音源に関しては最初に述べたように、基本的な原理を理解することと、マシン語によるドレミの演奏にターゲットを絞っていますので、音作りそのものに興味のある方は専門書による研究をお勧めします。ゲームにおいては、まず FM 音源による音楽演奏を、マシン語レベルで利用できるようになることが第一です。そこで、最も簡単な方法として ROM ルーチンを利用して、とにかく何か演奏をさせてみることにしましょう。演奏に際しては、V2 モードにした上で「NEW CMD」を実行する必要がありますが、「NEW CMD」はプログラムをアセンブルした後で行なってください。その理由は、拡張コマンド「CMD」が本体の方で使用されてしまい、アセンブル用としては使用できなくなってしまうからです。なお、この節にあるプログラムは 4、5 章で作るゲームには関係のない単独でのテスト・プログラムです。

この ROM ルーチンを利用するという方法は、プログラム (List 4-3) を見ても明らかのように、まるで BASIC の「CMD PLAY」と同じような感覚で、FM 音源と PSG を合わせた 6 音すべてを出すことができます。これなら、MML (Music Macro Language) さえ知っていれば、マシン語での音楽演奏も楽勝といえそうです。テストを実行しても、多分その期待は裏切られなかったと思います。…か、結論として、これをマシン語ゲームに利用するわけにはいかないのです。その理由は、これが BASIC で使用されているルーチンであるため、何とストップキーでブレイクされてしまうという、どうしようもない欠点 (マシン語ゲームにとって) があるからです。たとえ DI 命令を実行したとしても、ROM の中でストップキーのチェックがされている以上、これを止めることはできません。

List 4-3 ROM 内ルーチンによる FM 音源コントロール

10000		;***** List 4-3 *****	
	3926	MBIOS: EQU 3926H	;Music BIOS
		ORG 0D000H	
	D000	TEST: ;TEST	
	D000 1109D0	LD DE,VOICE	DE: 各音源データのテーブル・アドレス
	D003 0E01	LD C,1	1: 演奏のバレー・マーク
	D005 CD2639	CALL MBIOS	MBIOS: 44H に基づいて演奏
	D008 FF	RST 38H	
	D009	VOICE: ;VOICE	音源データ・テーブル
	D009 02	DB 2	CMD PLAY 文の # (音源) に相当する
	D00A 16D0	DW VD1	
	D00C 23D0	DW VD2	
	D00E 30D0	DW VD3	
	D010 30D0	DW VD4	各音源データのテーブル
	D012 48D0	DW VD5	
10190	D014 53D0	DW VD6	

D016		;VD1: ;Voice Data 1	——FM 音源 CH 1 の MML
D016	40315631	DB '@1V15CCEE GGG',0	
D01A	35434345		
D01E	45474747		
D022	00		
D023		;VD2: ;Voice Data 2	——FM 音源 CH 2 の MML
D023	40335631	DB '@3V15CCEE GGG',0	
D027	35434345		
D02B	45474747		
D02F	00		
D030		;VD3: ;Voice Data 3	——FM 音源 CH 3 の MML
D030	40375631	DB '@7V15GCGCGEC',0	
D034	35474347		
D038	43474543		
D03C	00		
D03D		;VD4: ;Voice Data 4	——SSG 音源 CH 1 の MML
D03D	56313543	DB 'V15CCCCCCCC',0	
D041	43434343		
D045	434300		
D048		;VD5: ;Voice Data 5	——SSG 音源 CH 2 の MML
D048	56313545	DB 'V15EEEEEEEE',0	
D04C	45454545		
D050	454500		
D053		;VD6: ;Voice Data 6	——SSG 音源 CH 3 の MML
D053	56313547	DB 'V15GGGGGGGG',0	
D057	47474747		
D05B	474700		



5. ミュージック…FM音源でハーブシコード

ROM ルーチンに頼れないとなると、シンセサイザーIC(ヤマハのYM-2203)を自分で直接コントロールするしかありません。つまり、このシンセサイザーICのFM音源部(以下、FM音源ICと略す)に対して、音色とか音階のデータを送るということです。むずかしそうな気がするかもしれませんが、そこには簡単なルールがありますから、素直にそれに従えさえすればいいのです。では、まずはルールの1として、音を出すための大まかな手順を理解してください。

- 1) FM音源ICに対し、音色データを送る
- 2) FM音源ICに対し、キー・オン(音を出すという合図)データを送る
- 3) FM音源ICに対し、音程データを送る
……………音が出る……………
- 4) 音の長さ分だけ、ウェイトをおく

- 5) FM音源ICに対し、キー・オフ(音を止めるという合図)データを送る

……………音が止まる……………

音色データは、音色の変更がない限りデータを送り直す必要はありませんから、実際に曲を演奏する場合は2)~5)を繰り返すということになります。いかにも簡単そうなルールの1)ですが、問題はこの「データを送る」という部分にあります。ウェイト以外はすべて「データを送る」になっていますが、FM音源ICのどこに、どんなデータを、どういう方法で、ということが具体的にまったく明記されていません。実は、これが本節最大のヤマ場であるルールの2というわけなのです。テーマが色々ありますから、1つ1つ順を追ってクリアしていくことにしましょう。

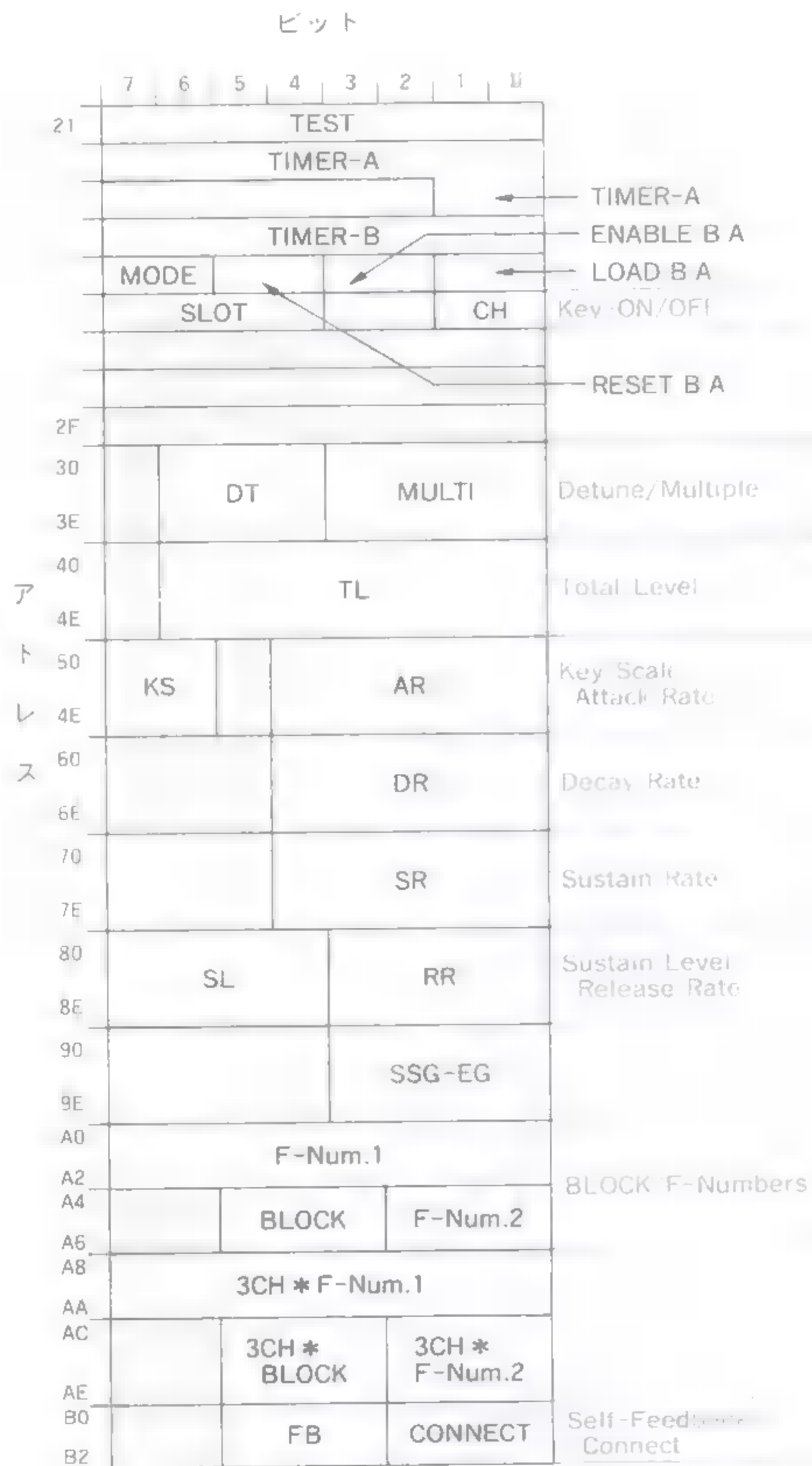
(1) FM音源ICのアドレス・マップ

まず、データの送り先ですが、FM音源ICのレジスタ・アドレスは図11のようになっています。コメントのある部分のアドレスが、ここで音を出すために必要なレジスタ(FM音源ICのメモリー)です。それ以外のレジスタに関しては、ここでは無視します。

この図を見ると、1つのアドレスで複数のパラメータを持っているものがあり、あまりわかりやすい構成とはいえません。しかし、ルールの1)にあるデータの内、キーのオン/オフと音程データの送り先は、確認できたと思います。そうすると、必然的に残りのレジスタ(コメントのないものは除

く)は、音色に関するレジスタということになります。つまり、めんどうなのは最初に音色を設定することで、音階を演奏するのは3つのレジスタにデータを送るだけでいいのです。ただし、FM音源は全部で3チャンネルありますから、3重奏をするためには、データもチャンネルごとに別々に送

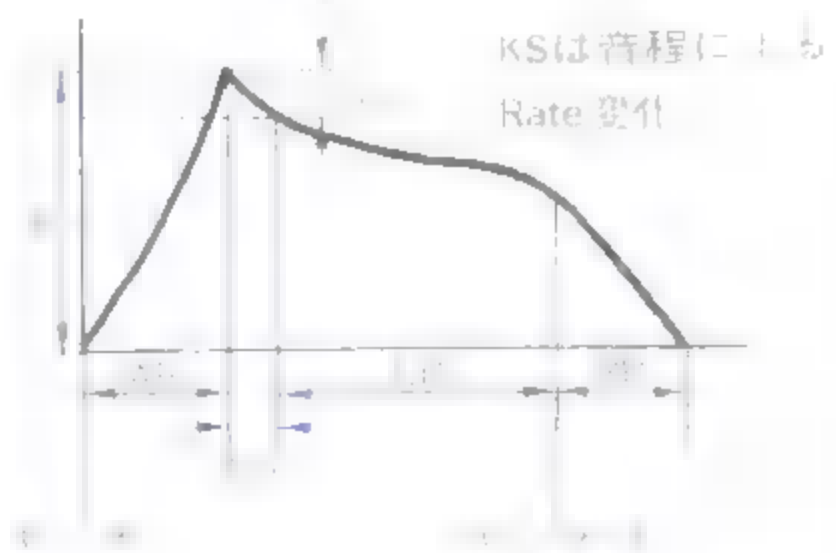
図 11



キー(オペレータ)のオン/オフ,
チャンネルの選択

音程のズレ、周波数比

エンベロープ・データ



オクターブ、音程データ

第1オペレータの変調度,
アルゴリズム

らなければなりません。

なお、各レジスタ・アドレスの意味については、ここでは概略だけしか書いてありませ

るので、詳しくは SR 本体についているマニュアルを読んでください。

(2) FM 音源 IC へ送るデータ

FM 音源 IC のレジスタ構造について、その概略は図 11 でつかめたと思いますが、実際にデータを送るにはチャンネルごとのアドレスとか、音色や音程のデータなどについて、もう少し詳しく知る必要があります。まず、各レジスタへ送るデータですが、これには「オペレータに対するデータ」と「チャンネルに対するデータ」があります。1つのチャンネルは4つのオペレータで構成されていますから、「オペレータに対するデータ」は1チャンネルにつき4つ要るということです。例えば、「DT/MULTI」に関するレジスタ・アドレスは30H～3EHですが、アドレスの内訳は下のようになっています。

アドレスの割り振り方は、他の項目についても同様です。また、「チャンネルに対するデータ」の場合は、各チャンネルについてデータは1つですから、アドレスは単純にチャンネル0～2の順に並んでいます。各パラメータの設定範囲は、図12に示されている通りですが、1つのアドレスを複数のパラメータで使用している場合には、それぞれのデータをいったん2進数に直してから所定のビットへ置き、改めている1つのデータにまとめる必要があります。

音程データは、11ビットを使って表現されていますから、データを送る時も上位(オクターブ・データ+F-Num.2)から順に送らなければなりません。

なお、具体的な音色データ(ピアノとかバイオリンなどの)は、システム・ディスクに入っている「mek」を走らせることにより、各パラメータの値が表示されますので、それを利用してください。ただし、「mek」ではパラメータの表現が少し違っており、

「SLOT」が「Operation Mask」、「TL」が「OL」、「CONNECT」が「Algorithm」にそれぞれ変わっています。また、LFO 効果については本来このシンセサイザー IC ではサポートされておらず、SR 本体側のソフトにより実現している付加機能です。ですから、どうしても LFO を使った音色にしたい場合は、音色の設定だけ BASIC で行なってください

図 12

『DT/MULTI』のアドレス内■

	チャンネル・0	チャンネル・1	チャンネル・2
オペレータ・1	30H	31H	32H
オペレータ・3	34H	35H	36H
オペレータ・2	38H	39H	3AH
オペレータ・4	3CH	3DH	3EH

図 13

各パラメータの範囲

パラメータ	設定範囲
SLOT (各オペレータのオン/オフ)	0 ~ 15
CH (チャンネル)	0 ~ 2
DT (音程のズレ)	-3 ~ 3
MULTI (周波数比)	(小) 0 ~ 15 (大)
TL (出力レベル)	(大) 0 ~ 127 (小)
KS (EGのRate変化)	(小) 0 ~ 3 (大)
AR (音の立ち上がりRate)	(長) 0 ~ 31 (短)
DR (SLまでのRate)	(長) 0 ~ 31 (短)
SR (SLからTL=0までのRate)	(長) 0 ~ 31 (短)
SL (DRの減衰量)	(小) 0 ~ 15 (大)
RR (キー・オフからTL=0までのRate)	(長) 0 ~ 15 (短)
F-Num.2+1 (音程データ)	[右図]
BLOCK (オクターブ)	(低) 0 ~ 7 (高)
FB (第1オペレータの変調幅)	(オフ) 0 ~ 7 (4π)
CONNECT (アルゴリズム)	0 ~ 7

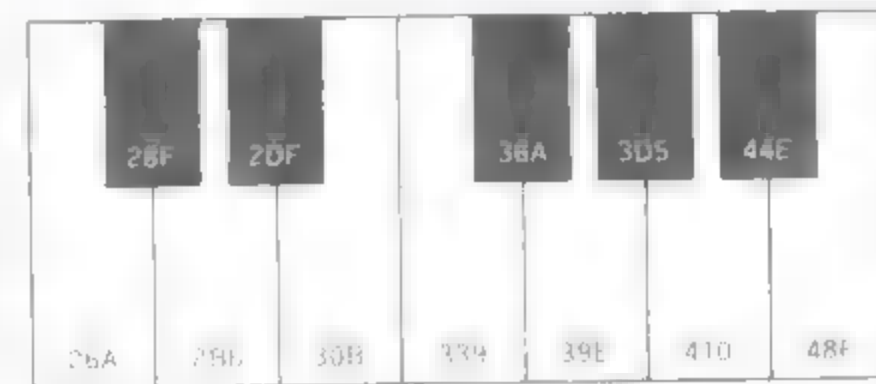
* Rateとは、時間を決めるための割合

音程データ……11ビットで表現される



F-Num. 2

F-Num. 1



(3) データの送り方

たくさんのレジスタ・アドレスを持ったシンセサイザーICですが、本体のメイン CPU とは I/O ポートの 44H と 45H だけで結ばれています。そのため、FM 音源 IC へデータを送るには、出力ポート 44H へまず送り先(レジスタ・アドレス)を出力し、次に出力ポート 45H へデータを出力する、という方法になっています。ここでもまたルールがあり、44H へ出力した後は 17 ステート(ステートはマシン語実行時間を数える単位)、45H へ出力した後は 83 ステート、それぞれ FM 音源 IC に対してウェイトをおく必要があります。ただし、45H へ出力した後のウェイトは、入力ポート 44H のビット 7 の値により、FM 音源 IC に対し出力 OK か否かの判断(ビット 7=0 なら OK)ができるようになっています。

以上が、FM 音源 IC をコントロールするための方法です。文章にすると、ルールだらけという感じがするかもしれませんがプログラムの方はデータが多いだけで、意外とスッキリしています。とりあえず、音色をハープシコードに設定し、当初の目的通

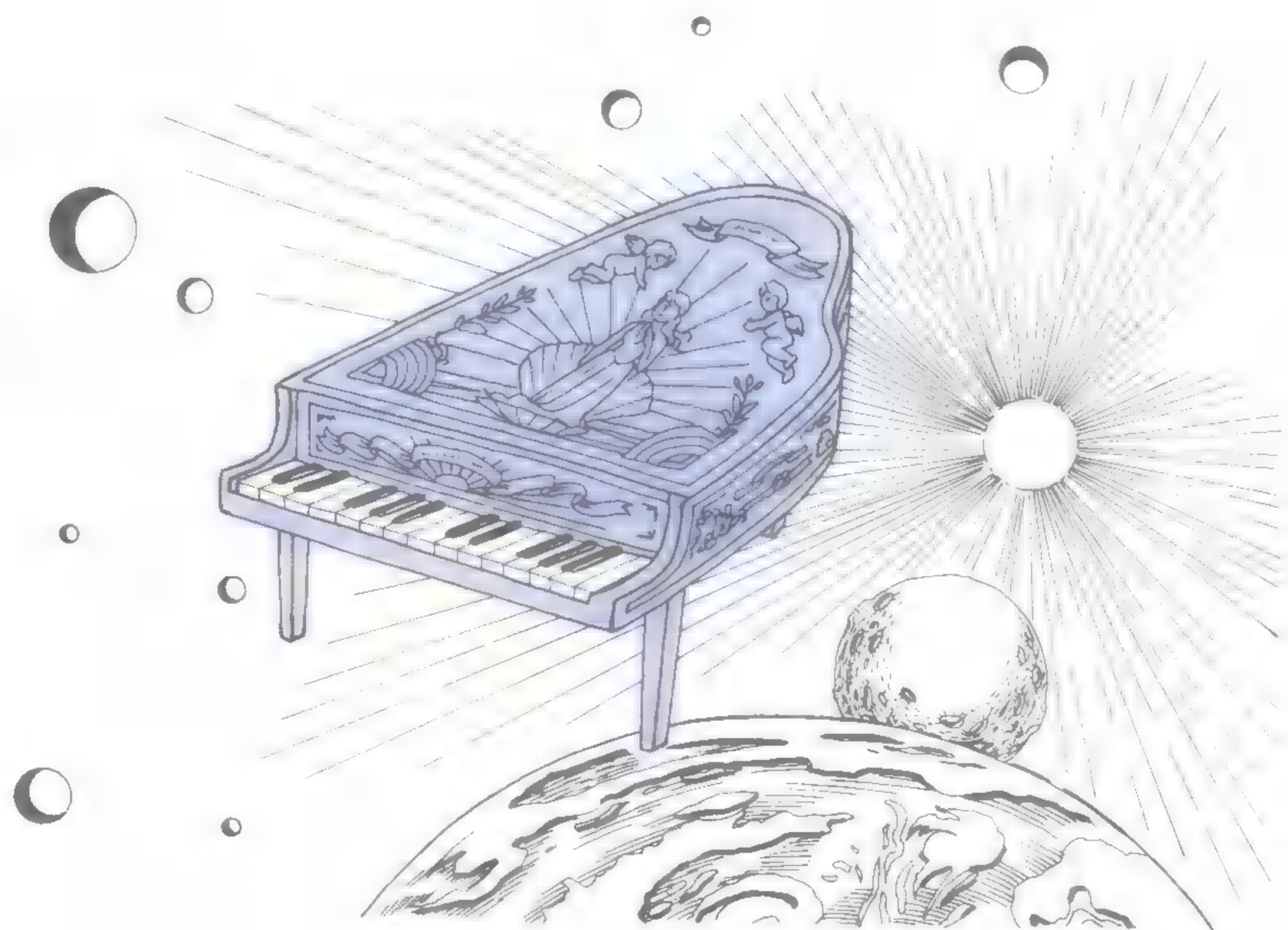
り「ドレミファソラシド」を演奏してみることにしましょう。

ここでは、FM 音源 3 チャンネル全部の音をハープシコードにしていますが、「mek」による別のデータでもテストしてみてください。ボリュームはマシン語で直

接実行した方が大きくなります。これは BASICにおけるボリュームの設定が少し低めになっているためです。また、いずれにしても音色を自分で直接コントロールできるようになったわけですから、メーカー指定の音色にこだわらずに、よりリアルな音を求めていってみてください。色々とデータを変えている内に、アッと驚くような「イイ音」に巡り会えるかもしれません。なお、このプログラムは出力ポートを直接操作しているので、実行に際して NEW

CMD』やモードの選択をする必要はまったくありません。つまり、アセンブラの『CMD』が何度でも使えるということであり、データを修正してテストをすることも簡単にできるわけです。やはり、『苦あれば楽あり』でしたね。

音楽というテーマは、まだまだ奥が深く、効果音、PSG、ミュージック割り込み、……等々、ページと時間に制限がなければ、もっともっと追求したいのですが、本書では残念ながらこの程度が限界のようです。ただ



し、読者の皆さんの希望が強ければ、また別の企画が立つかもしれません。とりあえず、本書はゲームマシン語のテーマが、まだまだ終わっていませんから(というより、

先の方が長〜い)、ここはFM音源に対する自信だけをつけたことにして、先へと進むことにいたしましょう。

List 4-4 FM音源によるハーブシコード

```

;***** List 4-4 *****
;
;          ORG 0D000H
;
TEST:  ;TEST
D000      LD B,75
D000 064B  LD HL,RDATA
D002 217ED0
D005      TLP1: ;Test Loop 1
D005 56      LD D,(HL)
D006 23      INC HL
D007 5E      LD E,(HL)
D008 CD1FD0  CALL OUT445
D00B 23      INC HL
D00C 10F7    DJNZ TLP1
;
D00E 2114D1  LD HL,SDATA
D011 0608    LD B,8
D013      TLP2: ;Test Loop 2
D013 CD30D0  CALL PLAY
D016 CD60D0  CALL WAIT
D019 CD6BD0  CALL KEYOFF
D01C 10F5    DJNZ TLP2
D01E FF      RST 38H
;
OUT445: ;OUT 44 & 45
D01F      IN A,(44H)
D01F DB44    AND 80H
D021 E680    JR NZ,OUT445
D023 20FA    LD A,D
D025 7A      OUT (44H),A
D026 D344    NOP
D028 00      NOP
D029 00      NOP
D02A 00      NOP
D02B 00      NOP
D02C 7B      LD A,E
D02D D345    OUT (45H),A
D02F C9      RET
;

```

```

D030      PLAY: ;PLAY music
D030 11F028 LD DE,28F0H
D033 CD1FD0 CALL OUT445
D036 11F128 LD DE,28F1H
D039 CD1FD0 CALL OUT445
D03C 11F228 LD DE,28F2H
D03F CD1FD0 CALL OUT445
;
D042 5E LD E,(HL)
D043 16A4 LD D,0A4H
D045 CD1FD0 CALL OUT445
D048 14 INC D
D049 CD1FD0 CALL OUT445
D04C 14 INC D
D04D CD1FD0 CALL OUT445
D050 23 INC HL
D051 5E LD E,(HL)
D052 16A0 LD D,0A0H
D054 CD1FD0 CALL OUT445
D057 14 INC D
D058 CD1FD0 CALL OUT445
D05B 14 INC D
D05C CD1FD0 CALL OUT445
D05F 23 INC HL
;
D060      WAIT: ;WAIT
D060 C5 PUSH BC
D061 019600 LD BC,150
D064      WTLP: ;Wait Loop
D064 10FE DJNZ WTLP
D066 0D DEC C
D067 20FB JR NZ,WTLP
D069 C1 POP BC
D06A C9 RET
;
D06B      KEYOFF: ;KEY OFF
D06B 110028 LD DE,2800H
D06E CD1FD0 CALL OUT445
D071 110128 LD DE,2801H
D074 CD1FD0 CALL OUT445
D077 110228 LD DE,2802H
D07A CD1FD0 CALL OUT445
D07D C9 RET
;
D07E      RDATA: ;Register DATA
D07E 300C310C DB 30H,12,31H,12,32H,12
D082 320C
D084 381F391F DB 38H,10H+15,39H,10H+15,3AH,10H+15
D088 3A1F
D08A 34013501 DB 34H,1,35H,1,36H,1
D08E 3601
D090 3C733D73 DB 3CH,70H+3,3DH,70H+3,3EH,70H+3
D094 3E73

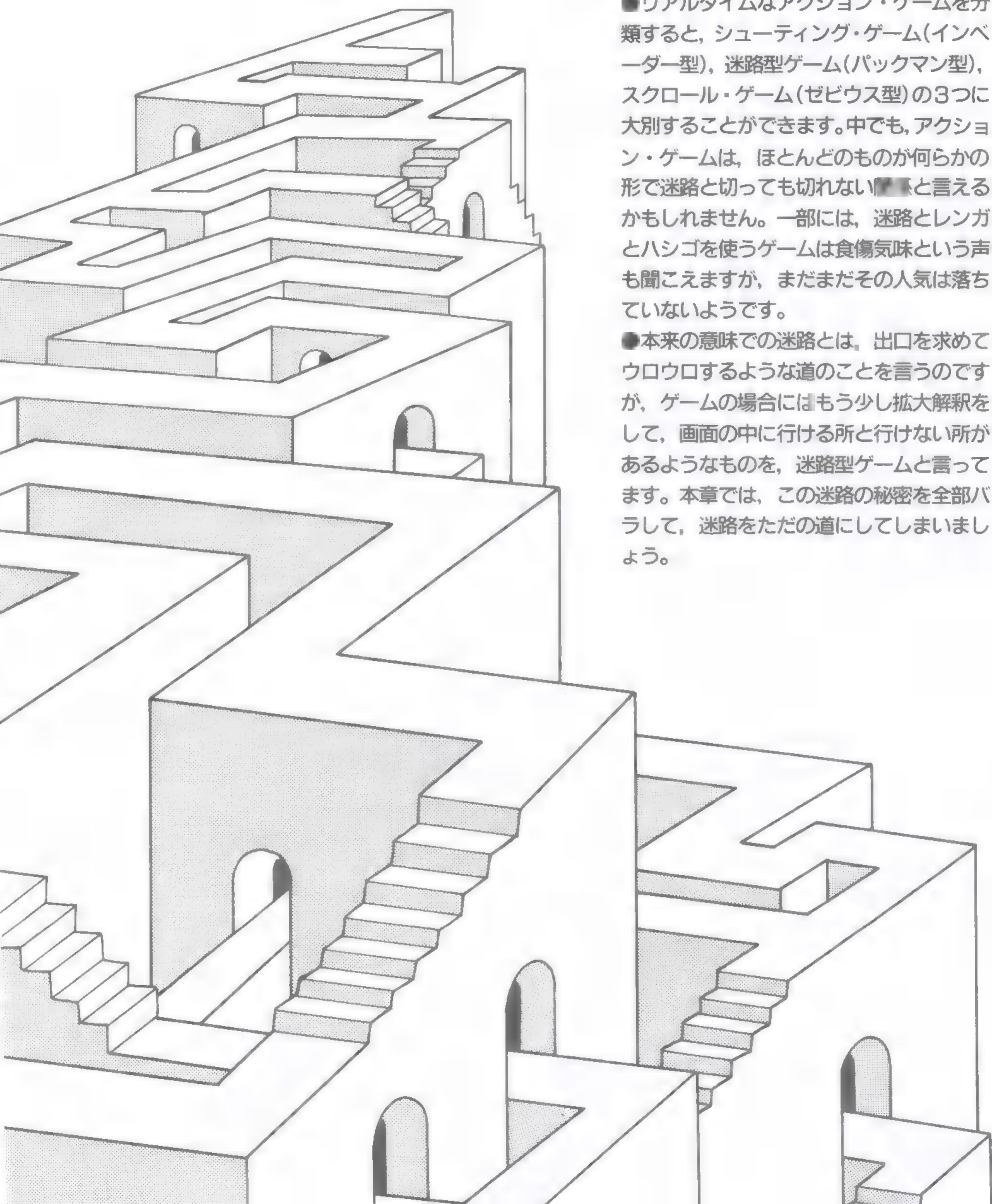
```


D096	40204120	DB	40H, 32, 41H, 32, 42H, 32	
D09A	4220			
D09C	48394939	DB	48H, 57, 49H, 57, 4AH, 57	レジスタ・アドレス
D0A0	4A39			
D0A2	441E451E	DB	44H, 30, 45H, 30, 46H, 30	
D0A6	461E			
D0A8	4C004D00	DB	4CH, 4DH, 0, 4EH, 0	
D0AC	4E00			
D0AE	501F511F	DB	50H, 31, 51H, 31, 52H, 31	レジスタ・アドレスと送るデータ
D0B2	521F			
D0B4	58DF59DF	DB	58H, 0C0H+31, 59H, 0C0H+31, 5AH, 0C0H+31	
D0B8	5ADF			
D0BA	541F551F	DB	54H, 31, 55H, 31, 56H, 31	
D0BE	561F			
D0C0	5C9F5D9F	DB	5CH, 80H+31, 5DH, 80H+31, 5EH, 80H+31	
D0C4	5E9F			
D0C6	600C610C	DB	60H, 12, 61H, 12, 62H, 12	
D0CA	620C			
D0CC	68026902	DB	68H, 2, 69H, 2, 6AH, 2	
D0D0	6A02			
D0D2	640C650C	DB	64H, 12, 65H, 12, 66H, 12	レジスタ・アドレスと送るデータ
D0D6	660C			
D0D8	6C056D05	DB	6CH, 5, 6DH, 5, 6EH, 5	
D0DC	6E05			
D0DE	70047104	DB	70H, 4, 71H, 4, 72H, 4	
D0E2	7204			
D0E4	78047904	DB	78H, 4, 79H, 4, 7AH, 4	
D0E8	7A04			
D0EA	74047504	DB	74H, 4, 75H, 4, 76H, 4	
D0EE	7604			
D0F0	7C077D07	DB	7CH, 7, 7DH, 7, 7EH, 7	
D0F4	7E07			
D0F6	801A811A	DB	80H, 10H+10, 81H, 10H+10, 82H, 10H+10	レジスタ・アドレスと送るデータ
D0FA	821A			
D0FC	88F689F6	DB	88H, 0F0H+6, 89H, 0F0H+6, 8AH, 0F0H+6	
D100	8AF6			
D102	84068506	DB	84H, 6, 85H, 6, 86H, 6	
D106	8606			
D108	8C278D27	DB	8CH, 20H+7, 8DH, 20H+7, 8EH, 20H+7	
D10C	8E27			
D10E	B03AB13A	DB	0B0H, 38H+2, 0B1H, 38H+2, 0B2H, 38H+2	
D112	B23A			
D114		SDATA: ; Sound DATA		Self-Feedback/Connection のレジスタ・アドレスと送るデータ
D114	1A6A	DB	18H+2, 6AH	1 オクターフ目 ト
D116	1AB6	DB	18H+2, 0B6H	1 オクターフ目 レ
D118	1B08	DB	18H+3, 0BH	1 オクターフ目 ミ
D11A	1B39	DB	18H+3, 39H	1 オクターフ目 フ
D11C	1B9E	DB	18H+3, 9EH	1 オクターフ目 シ
D11E	1C10	DB	18H+4, 10H	1 オクターフ目 ラ
D120	1C8F	DB	18H+4, 8FH	1 オクターフ目 シ
D122	226A	DB	20H+2, 6AH	5 オクターフ目 ト

● 迷路型ゲーム

1. 座標データ…行ける? 行けない?
2. 圧縮…座標データのデータ量
3. キー入力…操作性の向上
4. 追跡…サアー, 追いかけてよう!
5. 完成…メッセージや音を付ける





●リアルタイムなアクション・ゲームを分類すると、シューティング・ゲーム(インベダー型)、迷路型ゲーム(パックマン型)、スクロール・ゲーム(ゼビウス型)の3つに大別することができます。中でも、アクション・ゲームは、ほとんどのものが何らかの形で迷路と切っても切れない関係と言えるかもしれません。一部には、迷路とレンガとハシゴを使うゲームは食傷気味という声も聞こえますが、まだまだその人気は落ちていないようです。

●本来の意味での迷路とは、出口を求めてウロウロするような道のことを言うのですが、ゲームの場合にはもう少し拡大解釈をして、画面の中に行ける所と行けない所があるようなものを、迷路型ゲームと言っています。本章では、この迷路の秘密を全部バラして、迷路をただの道にしてしましましょう。

1. 座標データ…行ける? 行けない?

よく遊園地などに、ガラスで囲まれた部屋がたくさんある迷路があります。つまり、どこが出口かわからなくして楽しむのですが、もし床の部分に出口を示す矢印が描いてあったらどうでしょうか。楽しくはない代わりに、子供でも出口を間違えることはなくなりますね。これは、迷路を壁(ここではガラス)で捕えるのではなく、矢印によって判断させるようにすることですから、壁は視覚上の単なる飾りに過ぎなくなってしまうわけです。

迷路の中で行ける所と行けない所を判断する方法にこの考え方を取り入れようというのです。わかりやすくいうと、画面では壁が迷路を作っているように見えても、現実には壁ではなく矢印の方向によりパターンが動くようにするのです。

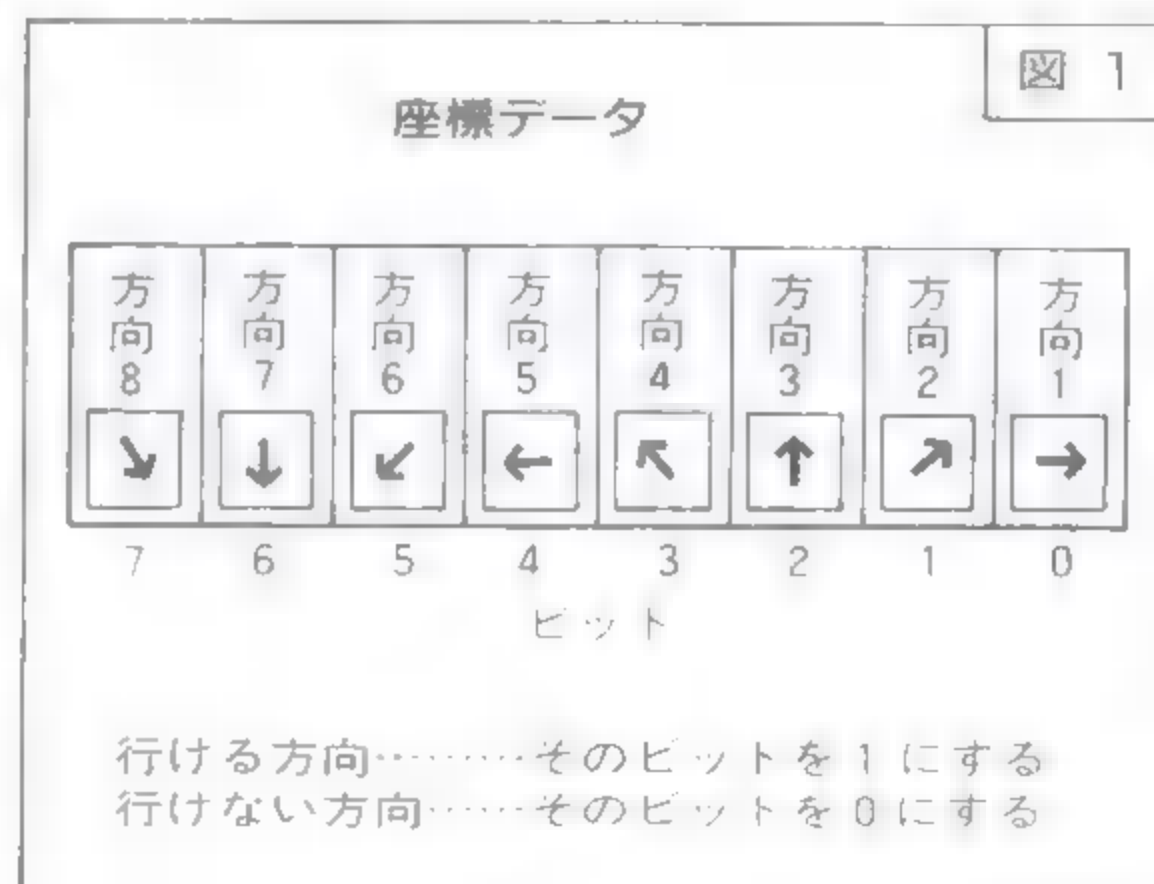
もちろん、実際のゲーム画面に矢印を描くわけにはいきませんから、矢印はそのまま別の場所に記憶しておくことになります。そして、パターンが移動できるすべてのゲーム座標にこの矢印を置き、1コマ移動するたびにそれをチェックするのです。こうすれば、パターンの位置から行ける所と行けない所の判断が、簡単にできるわけです。

何だか、わかったような、わからないような変な気がするかもしれませんが、もう少し具体的に考えてみましょう。矢印を置

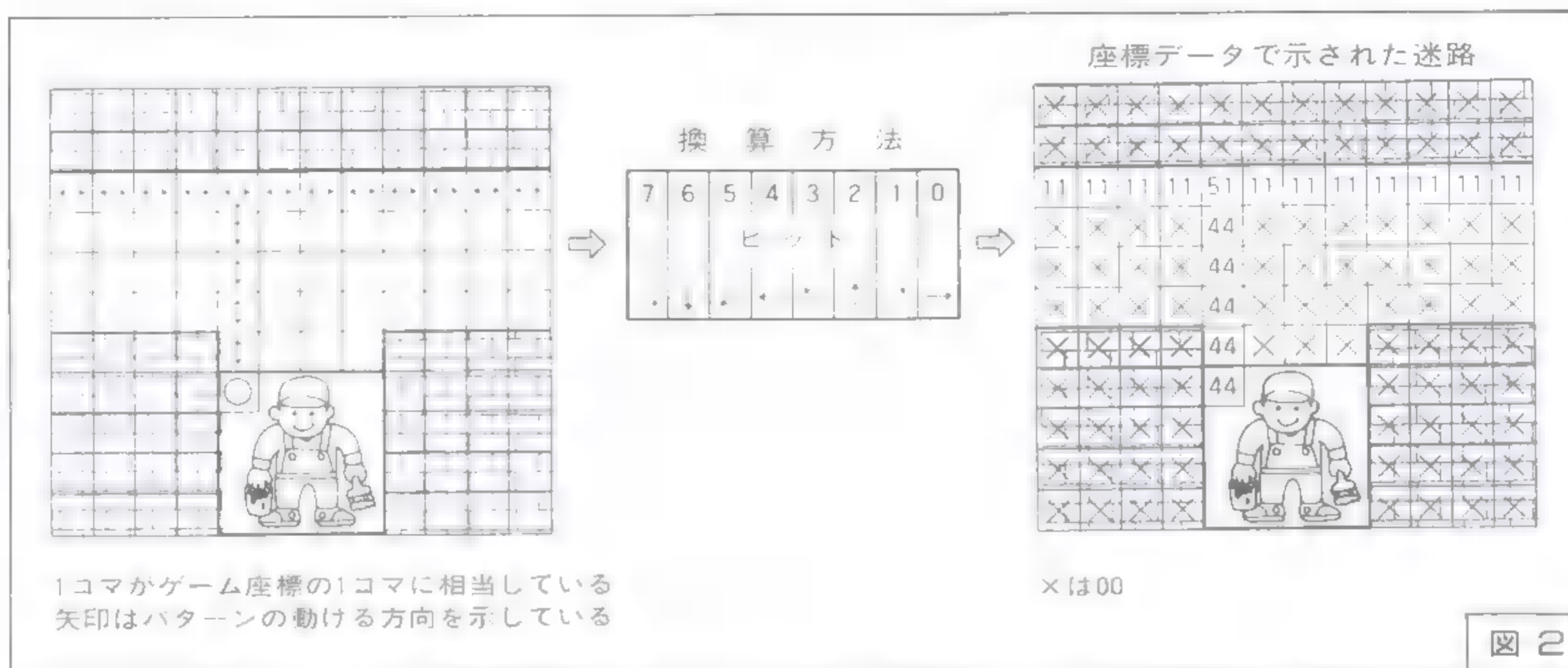
く場所とはメモリしかありません。メモリに置くためには数値でなければならないのは当然のことです。そこで、まず矢印を数値に変換する必要が生じてきます。矢印とは、すなわち行ける方向のことですから、全部で8方向分あります。一方、メモリに置ける数値も、1バイトつまり8ビットですから、1ビットごとに1つの方向を表わすようにし、行ける場合は1、行けない場合は0とすれば、その座標の矢印をすべて表現できることになります。

このようにして数値に変換された矢印のことを、その座標が持っているデータということで、座標データ*と呼ぶことにします。そして、ここでは矢印と座標データとの関係を下図のように設定して、矢印を数値に変換しています。

この座標データは、パターンを移動させ



*座標データは仮想 V-RAM と呼ばれています。



る前に読み出して、その方向にあたるビットをチェックすればいいだけです。利用方法も非常に簡単で確実です。また、どんなに複雑な迷路でも、これさえあれば作れるのですから迷路の必需品ともいえます。では、上図を見て下さい。

この座標データには、このような方法以外にも色々な作り方が考えられます。例えば、壁の部分を0、道の部分を1、ハシゴは2、道に金塊が置いてあれば3…というようなデータでも迷路の判断ができるわけです。つまり、座標データの内容に関しては、利用するあなたのアイディア次第ということであり、上図に示した座標データはその中の1つの例に過ぎません。いずれにしても、迷路の判断には画面とは別に何らかの座標データを持つ必要があるのです。

一般に、どんなゲームであっても画面数が少ないと、面白さも欠けているように思われる傾向があります。そのため、最近は最低でも20面位の画面数が要求されますが、そのような場合このままの座標データ

では、メモリが足りなくなってしまう。例えば、迷路画面のサイズをゲーム座標で(0,0)-(63,47)とすると、1画面につき実際に使用する座標データ数は

$$64 \times 48 = 3072 (= C00H)$$

となり、20面では0000H番地から使用しても、FFFFH番地までをすべて座標データで占領するということになります。

これでは、座標データを眺めながら、頭の中で勝手にゲームを想像して遊んでください、ということになってしまいます。しかし、現実には20面どころか100面、200面などという驚異的な面数を持つゲームがあるのです。この問題を解決するにはどうしたらいいか、それに対する答えは誰が考えても1つしかありません。それは、何とかして座標データそのものの数を減らす、ということです。そこで、次の節では座標データの圧縮をテーマにし、その可能性を追求してみましょう。

2. 圧縮…座標データのデータ量

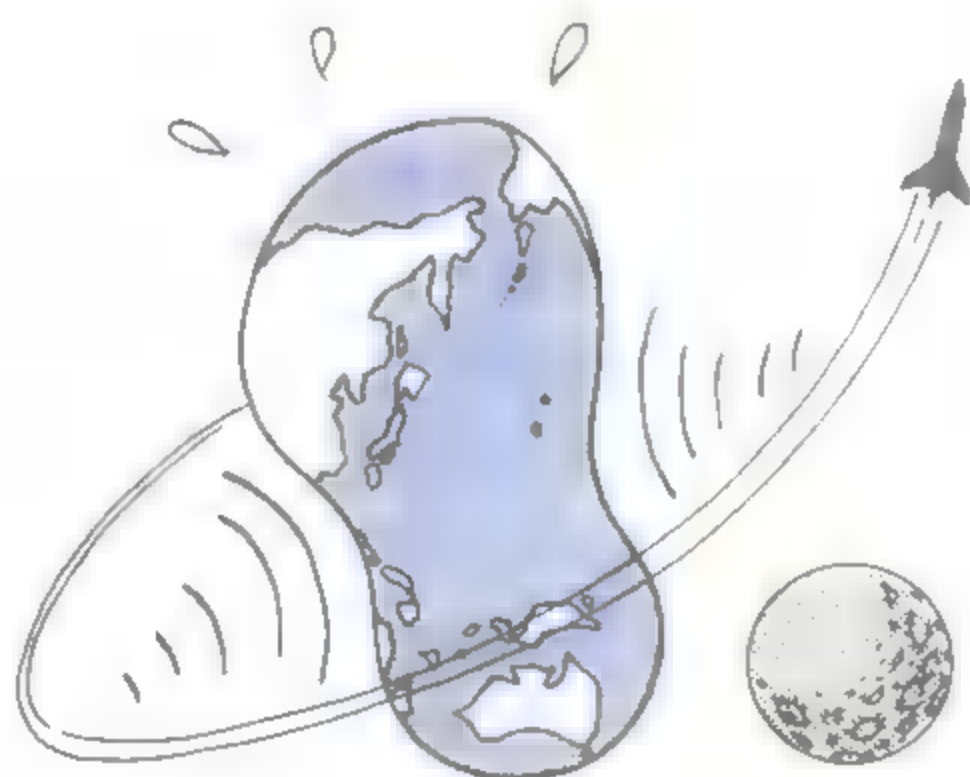
最近、ゲームの面数も異常と思えるほど多いモノが出てきています。面数だけを競うのはまったくナンセンスなことなのですが、1つだけ見習うべき点もあります。それは、面数を増やすためにデータ圧縮について非常に工夫を凝らしているということです。このデータ圧縮のテクニックを考え出すことに、ゲーム作者も多分に意義を感じていることと思います。もし、メモリが使い放題だとしたら、おそらく面数を増やす意欲は半減してしまうかもしれません。プログラムを作る面白さは、限られた条件の下で、いかにして自分の欲求を満たすかを考えることにあたりするのは私だけでしょうか……

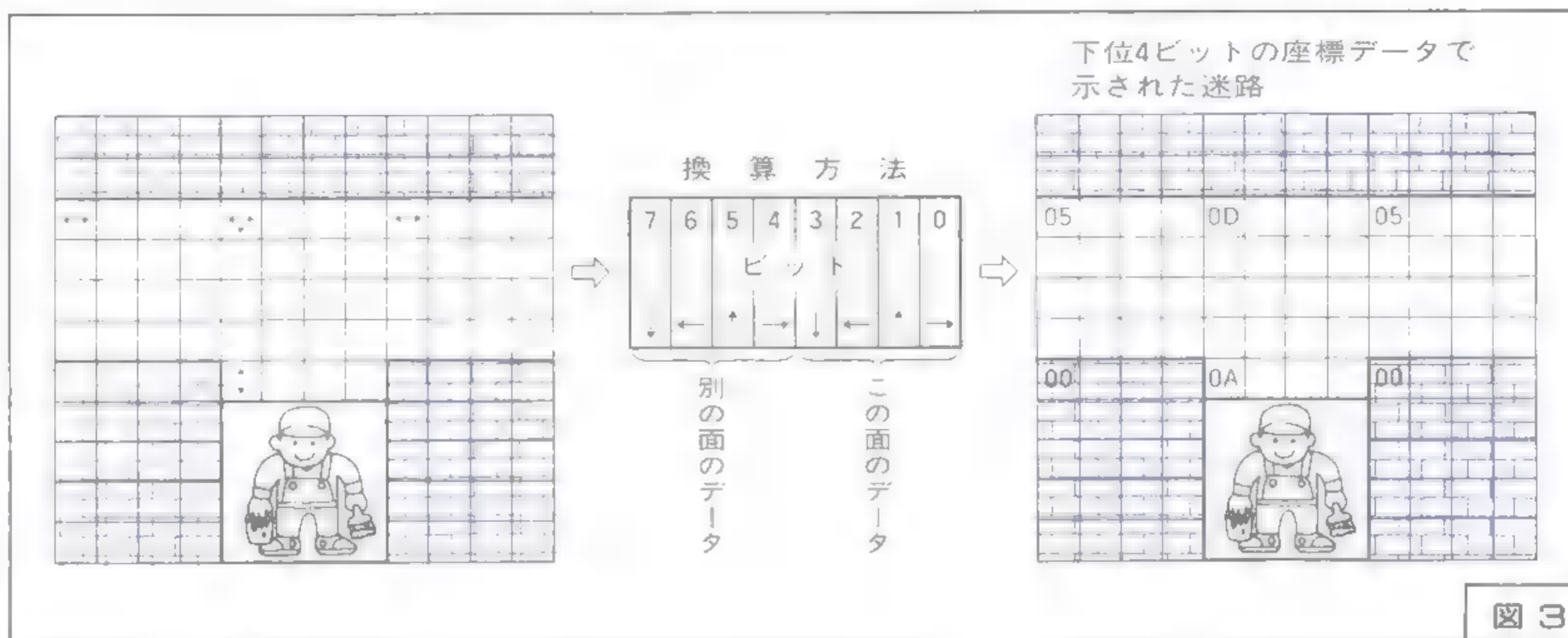
ゲームの説明などでよく40Kバイトを使っているとか、50Kバイトを使っているとか書かれていることがあります。これを、実際にプログラムとして使われている部分と、データとして使われている部分とに分

けてみると、7～8割はグラフィックを含めた各種のデータ・エリアとして使われているのが実情です。そこで、メモリを有効に使うためには、どうしてもデータを圧縮することを、考えなければならなくなってきました

圧縮というのは、必要があって初めて出てくるテーマですから、どこか発明に似た所があります。発明には「必要は発明の母」という有名な諺があり、いい発明は困った時こそ多く出るそうです。そして、発明をするには垂直思考的な発想ではなく、水平思考が大切であるといわれています。具体的な例でいうと、井戸を掘っていて岩に当たったとします。その時に、岩を取り除くとか、爆破しようというのは垂直思考で、別の所に穴を掘るという考えが水平思考なのです。一般に論理的な学問では、垂直思考的に問題を解いていきますが、発明においては水平思考が重要な発想法となっているそうです。

さて、問題になっている座標データを圧縮する方法ですが、ここでは2段階に分けて圧縮をかけてみます。まず第一段階では、パターンの方角変更は4コマ毎ということにして、座標データの数そのものを1/16にいきなり減らしてしまいます。4コマ毎に方向変更を制限するといっても、反対方向への変更は常に可能ですから、4コマを1ブロックとするキッチンとした迷路であれば、制限がないのとまったく同じことです。ただし、広場のような部分がある場合には、





方向変更が4コマ毎にしかできないため、多少動きがギクシャクします。次に、パターンの斜め移動を禁止して、上下左右だけの動きに制限します。これで、1つの座標データを上位4ビットと下位4ビットに分けて、別の面で使うことができるようになります。その結果、初期のものに比べると1/32のデータ量で済むことになったわけです。

上の図3を見ると、データ量は確かに大幅に少なくなっています。この方式による座標データであれば、200画面の迷路を作ることが可能であることも間違いありません。…が、これは実際には「圧縮されたデータ」とはいえないのです。というのは、この座標データはパターンの動きに制限を加えた結果生まれたものであり、データとしての基本的な構造は、図2と何ら変わりがないからです。本来、圧縮されたデータとはプログラムにより元の状態に戻すことができるはずですが、図3のデータはこれ自身がデ

ータであって、元に戻すべき姿はありません。そこで、方向変更は4コマ毎という条件は同じにして、次のような考え方で本格的な圧縮をかけてみます。なお、4コマ毎の方向変更ということは、壁、道、移動パターンをすべてゲーム座標で4×4コマのサイズに統一する、ということが前提となります。

1. 画面で壁の部分を1とする。
2. 画面で道の部分を0とする。
3. 横8ブロック(1ブロックはゲーム座標で4×4コマとする)分の壁と道を、連続する8ビット(1バイト)のデータとみなす。

この方法により作られたデータは、座標データではなく迷路の状態(壁か道)を表わしているので迷路データといえます。ゲーム座標で(0,0)-(63,47)の画面サイズを例にとって、図3と必要バイト数を比較してみましょう。

迷路データによる必要バイト数… $64/4/8 \times 48/4 = 24$

図3による必要バイト数…………… $64/4/2 \times 48/4 = 96$

これまでに比べ、更に 1/4 に圧縮されています。ただし、迷路データをこのまま座標データとして使うことはできませんから(処理速度を無視すれば可能)、これを矢印を表わす座標データに変換する必要があります。

この変換は、下図のように座標データを作ろうとするブロックの上下左右の状態を、迷路データで調べ、道であればその方向のビットを立てる、という作業を全部の道について行えば OK です。そのためには、1面分相当の座標データ・エリアは用意しておかなければなりませんが、面数が増えれば増えるほど、このデータ圧縮の効果も大きくなってきます。

なお、斜め移動はここでは無視していますが、上下左右に加えて、斜めの状態も調べればできるようになります。

さて、データ圧縮についての基本的な方針が理解できたところで、この 5 章で作る迷路ゲームに駒を進めることにいたしましょう。

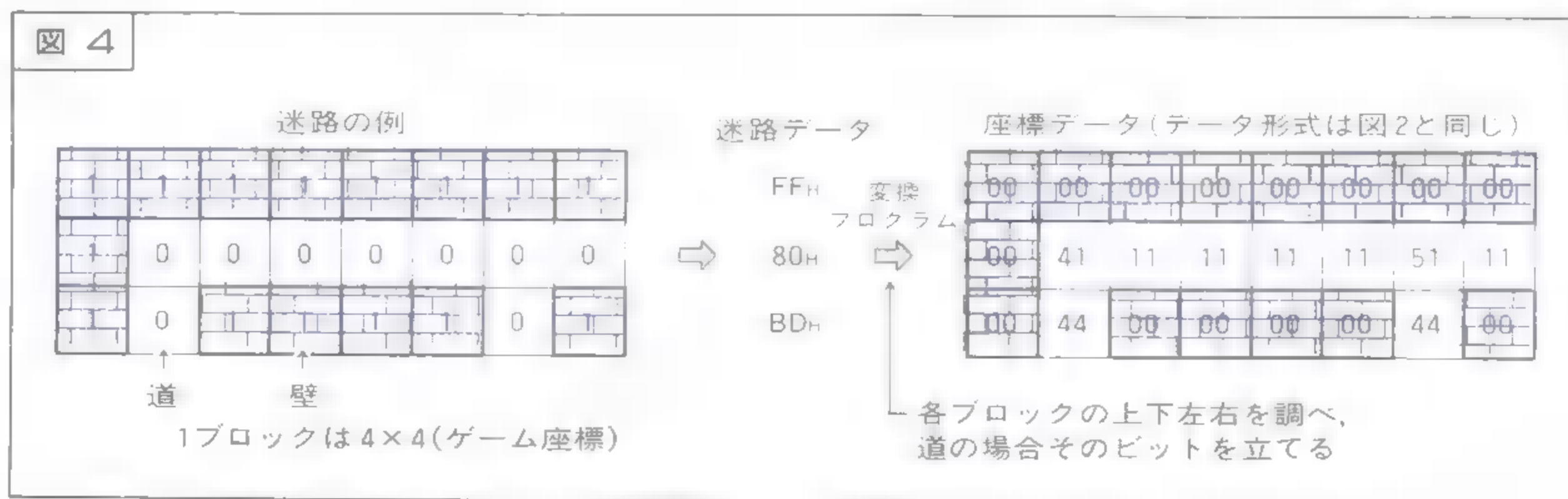
今回は、ゲームにも『ペンキ・ボーイ』という名前をつけてみました。主人公は、ヒデ君(名前の由来は、本人の希望もあり特

に秘す)という名の男の子で、道路をペイントするのが仕事です。道路は、色がすぐにハゲないように、特殊塗料で 7 層塗りをして、白くしなければなりません。しかし、例によってイジワルな敵がヒデ君の邪魔をしようと、迷路の中をウロウロしています…というのが、このゲームのイメージ・ストーリーです。

イメージの世界から一転して、現実に戻ります。このゲームの内容を具体的に見てみると、まず迷路内には 3 種類の敵と、主人公ヒデ君がいます。ゲーム・スタート時の道の色は黒ですが、ヒデ君が歩いた跡は青(1)から白(7)へパレット番号順に変化していきます。そして、すべての道が同一色になるとボーナス点が入り、すべての道が白になった時点でゲーム終了となります。スペースバーを押すと、道の色を変化させずに歩くことができ、また敵と衝突しても死なないものとします。

作成する面数は、練習ですから 1 面だけしかありません。以上がゲームの概略ですが、本節では迷路データから座標データへの変換、および画面への迷路表示までを行います。

図 4



方 向 番 号	
方向	16進数 2進数
→	40H 0 1 0 0 0 0 0 0
←	10H 0 0 0 1 0 0 0 0
↓	04H 0 0 0 0 0 1 0 0
↑	01H 0 0 0 0 0 0 0 1

入力キー	(入力ポート)
6	1 0 1 1 1 1 1 1 (00H)
4	1 1 1 0 1 1 1 1 (00H)
2	1 1 1 1 1 0 1 1 (00H)
8	1 1 1 1 1 1 1 0 (01H)

7	6	5	4	3	2	1	0
	→		←		↓		↑

新座標データ
1, 3, 5, 7は未使用

なお、この迷路ゲームでは方向を示す番号を、これまでのものと変えてあります。それは、キー入力で得られるビット(押されたことを示すビット)を、そのまま方向番号として表現したことです。もちろん、従来の方向番号で表現してもプログラムは組めるのですが、なるべく別のテクニックを取り入れようということと、キー入力で得たデータをそのまま方向番号として扱えば、少しは処理速度がアップするのではないかという理由からです。

実際には、わかりにくくなっただけと感ずるかもしれませんが、何事もチャレンジ精神とってください。慣れれば、その合

理性が理解できるはずですが。移動方向は上下左右だけで、方向番号は上図のようになります。当然、座標データの矢印も同じビットで表わすようになっていきます。

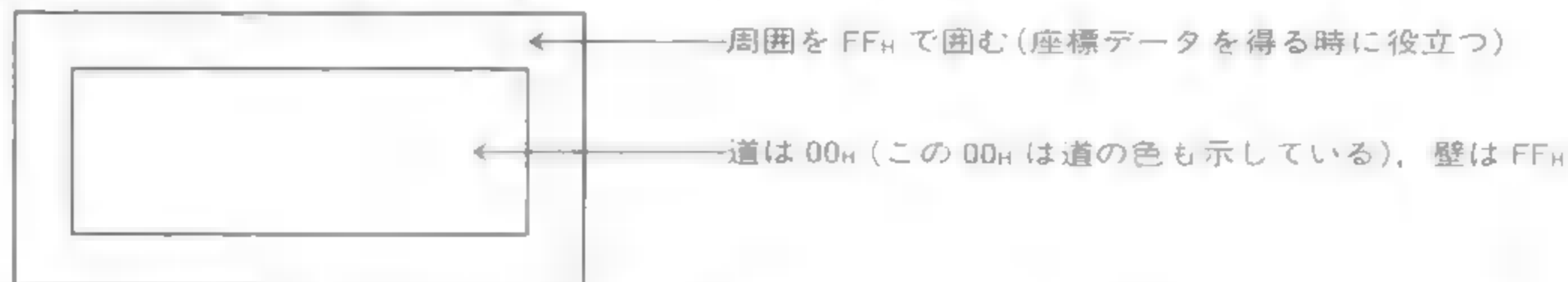
また、このゲームの性格上、座標データに相当するものがもう1つ必要です。それは道の色を示すデータで、パターンが移動する時にはその色で消去をしなければならないからです。

そのため、迷路データから座標データを得る時には、次ページの図6のような順序で変換をしていきます。

これで、プログラムへ入るための予備知識は完璧といえます。壁のパターン・データ

図 6

1. 迷路データを読む
2. 周囲を FFH で囲みながら道を 00H, 壁を FFH にする。…迷路データの拡大



3. 作成されたデータ (周囲の FFH は除く) から, 座標データを作る。道の上下左右をチェックし壁の場合は, 行ける方向がないので座標データを 00H とする。…座標データの作成
4. 迷路データから周囲の FFH を取り除き, 連続したデータとする。このデータが, 道の色を示すデータ・エリアとなり, 道の色が変化することによりデータはパレットコード通りに変化することになる。
5. 作成された連続したデータから迷路を描く。…画面の表示

例, 迷路サイズを 8×3 ブロックとした場合 (1 ブロックは座標データで 4×4 コマ)

1. 迷路データ

00H
76H
00H

- 2.

FF									
	00								

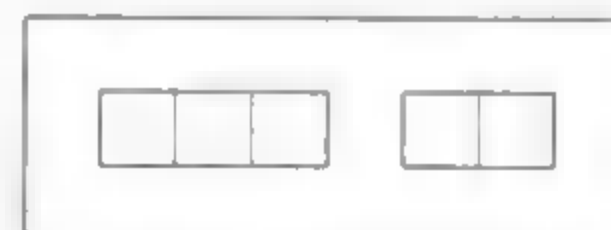
3. 座標データ

44	50	50	50	54	50	50	14
05	00	00	00	05	00	00	05
41	50	50	50	51	50	50	11

4. 仮想迷路

00	00	00	00	00	00	00	00
00	FF	FF	FF	00	FF	FF	00
00	00	00	00	00	00	00	00

5. 画面の表示



がありませんが, ここでのテストには特に支障ありませんので, プログラムを作成したら早速実行してみましょう。なお, 本章の各プログラムも 2, 3 章同様に次々と Merge していきます。また, ビープ音楽を利用しているので, List 4-1, 4-2 と Merge してください。最終的に迷路ゲームが完成する際には『MF-ASM』の限界に近い長さのプログ

ラムとなっていますので, コメントは掲載リスト中にあるもの以外一切入れないでください。さもないと, 最後にメモリ・オーバーになる可能性があります。

このプログラムの中で, 10410~10620 行に CLPTXY という消去ルーチンがありますが, この内容は 2, 3 章で使ったものと少し違っています。これは, このゲームのた

めに必要になったもので、消去する色をパレット番号で指定できるようにしています。そして、その面を表示するか消去するかはキャリーフラグを利用して、00HとFFHを作り出すように工夫してあります。また、座標データに関するルーチンは、図6を見ながらプログラムを追うと、わかりやすいと思います。迷路データ(22430～

22490 行の M2DATA)を変えれば、どんな迷路でも表示してくれますので、テストを試してみてください。ただし、このゲームでは主人公や敵の初期出現場所を、この迷路に合わせていますので、最後にはこのリスト通りのデータに戻しておかなければなりません。

List 5-1 座標データの作成と迷路の表示

10000		***** List 5-1-6 *****
		;
B600		STACK: EQU 0B600H ;STACK pointer
C000		VTOP: EQU 0C000H ;Vram TOP address
0050		HLEN: EQU 80 ;Horizontal LENGTH
		;
		ORG 0BF10H
		;
BF10		DISP: ;DISPlay
BF10 CD53C0		CALL XYADR
BF13 CD65C0		CALL PDADR
BF16 D35C		OUT (5CH),A
BF18 CD28BF		CALL BOX
BF1B D35D		OUT (5DH),A
BF1D CD28BF		CALL BOX
BF20 D35E		OUT (5EH),A
BF22 CD28BF		CALL BOX
BF25 D35F		OUT (5FH),A
BF27 C9		RET
		;
BF28	BOX: ;BOX	
BF28 ED7344BF		LD (LDSP+1),SP
BF2C 314C00		LD SP,HLEN-4
BF2F ED5B47BF		LD DE,(DISPAD)
BF33 01FF10		LD BC,10FFH
BF36	LOOP: ;LOOP	
BF36 EDA0		LDI
BF38 EDA0		LDI
BF3A EDA0		LDI
BF3C EDA0		LDI
BF3E EB		EX DE,HL
BF3F 39		ADD HL,SP
BF40 EB		EX DE,HL
BF41 10F3		DJNZ LOOP
BF43	LDSP: ;Load Stack Pointer	
BF43 310000		LD SP,0000
BF46 C9		RET

```

;
BF47 DISPAD: ; Address
BF47 DS 2
;
BF49 CLPTXY: ; Clear Pattern
BF49 2285BF LD (SIZE),HL
BF4C CD53C0 CALL XYADR
BF4F 3A87BF LD A,(COLOR)
BF52 0F RRCA
BF53 F5 PUSH AF
BF54 9F SBC A,A
BF55 D35C OUT (5CH),A
BF57 CD6EBF CALL ERBOX
BF5A F1 POP AF
BF5B 0F RRCA
BF5C F5 PUSH AF
BF5D 9F SBC A,A
BF5E D35D OUT (5DH),A
BF60 CD6EBF CALL ERBOX
BF63 F1 POP AF
BF64 0F RRCA
BF65 9F SBC A,A
BF66 D35E OUT (5EH),A
BF68 CD6EBF CALL ERBOX
BF6B D35F OUT (5FH),A
BF6D C9 RET
;
BF6E ERBOX: ; Erase Box
BF6E 2A47BF LD HL,(DISPAD)
BF71 115000 LD DE,HLEN
BF74 ED4B85BF LD BC,(SIZE)
BF78 ERL1: ; Erase Loop 1
BF78 C5 PUSH BC
BF79 E5 PUSH HL
BF7A ERL2: ; Erase Loop 2
BF7A 77 LD (HL),A
BF7B 23 INC HL
BF7C 10FC DJNZ ERL2
BF7E E1 POP HL
BF7F 19 ADD HL,DE
BF80 C1 POP BC
BF81 0D DEC C
BF82 20F4 JR NZ,ERL1
BF84 C9 RET
;
BF85 SIZE: ; SIZE
BF85 DS 2
BF87 COLOR: ; COLOR
BF87 DS 1
;
E6C2 PORT31:EQU 0E6C2H ;data 1 out port PORT 31
;
BF88 CLS: ; Clear Screen
BF88 3AC2E6 LD A,(PORT31)
BF8B E6F7 AND 0F7H
BF8D D331 OUT (31H),A
BF8F D35C OUT (5CH),A
BF91 CDA6BF CALL ACLS
BF94 D35D OUT (5DH),A
BF96 CDA6BF CALL ACLS
BF99 D35E OUT (5EH),A

```



```

BF98 CDA6BF      CALL ACLS
BF9E D35F        OUT (5FH),A
BFA0 3AC2E6      LD A,(PORT31)
BFA3 D331        OUT (31H),A
BFA5 C9          RET
BFA6             ACLS: ;All CLS
BFA6 2100C0      LD HL,VTOP
BFA9 1101C0      LD DE,VTOP+1
BFAC 017F3E      LD BC,3E7FH
BFAF 3600        LD (HL),0
BFB1 EDB0        LDIR
BFB3 C9          RET
;
;***** List 5-1-N *****
;
C053             XYADR: ;XY to Address
C053 68          LD L,B
C054 2600        LD H,0
C056 29          ADD HL,HL
C057 29          ADD HL,HL
C058 29          ADD HL,HL
C059 29          ADD HL,HL
C05A 29          ADD HL,HL
C05B 29          ADD HL,HL
C05C 09          ADD HL,BC
C05D 0100C0      LD BC,VTOP
C060 09          ADD HL,BC
C061 2247BF      LD (DISPAD),HL
C064 C9          RET
;
C065             PDADR: ;Pattern Data Address
C065 6F          LD L,A
C066 2600        LD H,0
C068 29          ADD HL,HL
C069 1172C0      LD DE,PDBASE
C06C 19          ADD HL,DE
C06D 7E          LD A,(HL)
C06E 23          INC HL
C06F 66          LD H,(HL)
C070 6F          LD L,A
C071 C9          RET
;
C072             PDBASE: ;Pattern Data Base address
C072 00B6C0B6    DW 0B600H,0B6C0H
C076 80B740B8    DW 0B780H,0B840H
C07A 00B9C0B9    DW 0B900H,0B9C0H
C07E 80BA40BB    DW 0BA80H,0BB40H
C082 00BC        DW 0BC00H
;
0010             IMZX: EQU 16 ;Image Maze X
000C             IMZY: EQU 12 ;Image Maze Y
0001             DU: EQU 01H ;Direction UP
0004             DD: EQU 04H ;Direction Down
0010             DL: EQU 10H ;Direction Left
0040             DR: EQU 40H ;Direction Right
;
C084             MZDATA: ;Maze DATA
C084 00006A56    DB 00H,00H,6AH,56H
C088 6A560A50    DB 6AH,56H,0AH,50H
0246 C08C 781E0240 DB 78H,1EH,02H,40H

```

画面ノ

変数

迷路のサイズ 横…16ブロック 縦…12ブロック

方向番号のラベル化(これまでと違う)

迷路データ

```

22470 C090 5E7A1008      DB  5EH,7AH,10H,08H
      C094 75AE05A0      DB  75H,0AEH,05H,0A0H
      C098 6C360180      DB  6CH,36H,01H,80H

      ;
      IMAZE:  ;Image MAZE          ——仮想迷路
      C09C      DS  252
      AMAZE:  ;Arrow MAZE          ——矢印迷路
      C198      DS  192

      ;
      MKIMZ:  ;Make Image maze Loop 2  ——仮想迷路を作る
      C258      LD  HL,IMAZE
      C258 219CC0      LD  DE,MZDATA
      C258 1184C0      LD  B,IMZX+2
      C25E 0612

      MILP1:  ;Make Image maze Loop 1  ——仮想迷路の1段目
      C260      LD  (HL),0FFH
      C260 36FF      INC  HL
      C262 23      DJNZ MILP1
      C263 10FB      LD  C,IMZY
      C265 0E0C      MILP2:  ;Make Image MaZe
      C267      LD  (HL),0FFH
      C267 36FF      INC  HL
      C269 23      LD  B,2
      C26A 0602      MILP3:  ;Make Image maze Loop 3
      C26C      PUSH BC
      C26C C5      LD  A,(DE)
      C26D 1A      INC  DE
      C26E 13      LD  B,8
      C26F 0608      MILP4:  ;Make Image maze Loop 4
      C271      LD  C,0
      C271 0E00      RLCA
      C273 07      JR  NC,MILP5
      C274 3001      DEC  C
      C276 0D      MILP5:  ;Make Image maze Loop 5
      C277      LD  (HL),C
      C277 71      INC  HL
      C278 23      DJNZ MILP4
      C279 10F6      POP  BC
      C27B C1      DJNZ MILP3
      C27C 10EE      LD  (HL),0FFH
      C27E 36FF      INC  HL
      C280 23      DEC  C
      C281 0D      JR  NZ,MILP2
      C282 20E3      LD  B,IMZX+2
      C284 0612      MILP6:  ;Make Image maze Loop 6
      C286      LD  (HL),0FFH
      C286 36FF      INC  HL
      C288 23      DJNZ MILP6
      C289 10FB      RET
      C28B C9

      ;
      MKAMZ:  ;Make Arrow MaZe          ——矢印迷路を作る
      C28C      LD  HL,AMAZE
      C28C 2198C1      LD  DE,IMAZE+IMZX+3
      C28F 11AFC0      LD  C,IMZY

      MALP1:  ;Make Arrow maze Loop 1
      C294      LD  B,IMZX
      C294 0610      MALP2:  ;Make Arrow maze Loop 2
      C296      PUSH BC
      C296 C5      LD  (HL),0
      C297 3600      LD  A,(DE)
      C299 1A      OR  A
      C29A B7      JR  NZ,NPOS
      C29B 2035
23070

```

——仮想迷路
 ——矢印迷路
 ——仮想迷路を作る
 ——仮想迷路の1段目
 ——仮想迷路の最下段
 ——矢印迷路を作る
 ——矢印迷路の1段目
 ——矢印迷路の2段目
 ——矢印迷路の3段目
 ——矢印迷路の4段目
 ——矢印迷路の5段目
 ——矢印迷路の6段目
 ——矢印迷路の7段目
 ——矢印迷路の8段目
 ——矢印迷路の9段目
 ——矢印迷路の10段目
 ——矢印迷路の11段目
 ——矢印迷路の12段目
 ——矢印迷路の13段目
 ——矢印迷路の14段目
 ——矢印迷路の15段目
 ——矢印迷路の16段目
 ——矢印迷路の17段目
 ——矢印迷路の18段目
 ——矢印迷路の19段目
 ——矢印迷路の20段目
 ——矢印迷路の21段目
 ——矢印迷路の22段目
 ——矢印迷路の23段目
 ——矢印迷路の24段目
 ——矢印迷路の25段目
 ——矢印迷路の26段目
 ——矢印迷路の27段目
 ——矢印迷路の28段目
 ——矢印迷路の29段目
 ——矢印迷路の30段目
 ——矢印迷路の31段目
 ——矢印迷路の32段目
 ——矢印迷路の33段目
 ——矢印迷路の34段目
 ——矢印迷路の35段目
 ——矢印迷路の36段目
 ——矢印迷路の37段目
 ——矢印迷路の38段目
 ——矢印迷路の39段目
 ——矢印迷路の40段目
 ——矢印迷路の41段目
 ——矢印迷路の42段目
 ——矢印迷路の43段目
 ——矢印迷路の44段目
 ——矢印迷路の45段目
 ——矢印迷路の46段目
 ——矢印迷路の47段目
 ——矢印迷路の48段目
 ——矢印迷路の49段目
 ——矢印迷路の50段目
 ——矢印迷路の51段目
 ——矢印迷路の52段目
 ——矢印迷路の53段目
 ——矢印迷路の54段目
 ——矢印迷路の55段目
 ——矢印迷路の56段目
 ——矢印迷路の57段目
 ——矢印迷路の58段目
 ——矢印迷路の59段目
 ——矢印迷路の60段目
 ——矢印迷路の61段目
 ——矢印迷路の62段目
 ——矢印迷路の63段目
 ——矢印迷路の64段目
 ——矢印迷路の65段目
 ——矢印迷路の66段目
 ——矢印迷路の67段目
 ——矢印迷路の68段目
 ——矢印迷路の69段目
 ——矢印迷路の70段目
 ——矢印迷路の71段目
 ——矢印迷路の72段目
 ——矢印迷路の73段目
 ——矢印迷路の74段目
 ——矢印迷路の75段目
 ——矢印迷路の76段目
 ——矢印迷路の77段目
 ——矢印迷路の78段目
 ——矢印迷路の79段目
 ——矢印迷路の80段目
 ——矢印迷路の81段目
 ——矢印迷路の82段目
 ——矢印迷路の83段目
 ——矢印迷路の84段目
 ——矢印迷路の85段目
 ——矢印迷路の86段目
 ——矢印迷路の87段目
 ——矢印迷路の88段目
 ——矢印迷路の89段目
 ——矢印迷路の90段目
 ——矢印迷路の91段目
 ——矢印迷路の92段目
 ——矢印迷路の93段目
 ——矢印迷路の94段目
 ——矢印迷路の95段目
 ——矢印迷路の96段目
 ——矢印迷路の97段目
 ——矢印迷路の98段目
 ——矢印迷路の99段目
 ——矢印迷路の100段目

C29D 1B	DEC DE	
C29E 1A	LD A,(DE)	← 仮想的迷路データ
C29F 13	INC DE	
C2A0 B7	OR A	
C2A1 2004	JR NZ,ARCKR	
C2A3 3E10	LD A,DL	A ← 迷路データの左方向
C2A5 86	ADD A,(HL)	
C2A6 77	LD (HL),A	
C2A7	ARCKR: ;Arrow Check Right	
C2A7 13	INC DE	← 右側の仮想迷路
C2A8 1A	LD A,(DE)	
C2A9 B7	OR A	
C2AA 2004	JR NZ,ARCKU	← 右側が壁なら、FLOCK
C2AC 3E40	LD A,DR	A ← 迷路データの右方向
C2AE 86	ADD A,(HL)	
C2AF 77	LD (HL),A	← 右側が壁なら、FLOCK
C2B0	ARCKU: ;Arrow Check Up	
C2B0 01EDFF	LD BC,-IMZX-3	
C2B3 EB	EX DE,HL	← 上側の横サイズ
C2B4 09	ADD HL,BC	
C2B5 EB	EX DE,HL	
C2B6 1A	LD A,(DE)	← 上側の縦サイズ
C2B7 B7	OR A	← 上側が壁なら、FLOCK
C2B8 2004	JR NZ,ARCKD	
C2BA 3E01	LD A,DU	A ← 迷路データの上方向
C2BC 86	ADD A,(HL)	
C2BD 77	LD (HL),A	
C2BE	ARCKD: ;Arrow Check Down	
C2BE 012400	LD BC,IMZX+IMZX+4	← 下側の横サイズ × 2 + 4
C2C1 EB	EX DE,HL	
C2C2 09	ADD HL,BC	
C2C3 EB	EX DE,HL	
C2C4 1A	LD A,(DE)	← 下側の仮想迷路データ
C2C5 B7	OR A	← 下側が壁なら、FLOCK
C2C6 2004	JR NZ,MAPOS	
C2C8 3E04	LD A,DD	A ← 迷路データの前方
C2CA 86	ADD A,(HL)	
C2CB 77	LD (HL),A	← 前方が壁なら、FLOCK
C2CC	MAPOS: ;Make Arrow Position	
C2CC 01EEFF	LD BC,-IMZX-2	
C2CF EB	EX DE,HL	← 前方の縦サイズ
C2D0 09	ADD HL,BC	← 前方の縦サイズ × 2 + 4
C2D1 EB	EX DE,HL	
C2D2	NPOS: ;Next Position	
C2D2 13	INC DE	← 仮想迷路データを+1する
C2D3 23	INC HL	← 仮想迷路データを+1する
C2D4 C1	POP BC	← 仮想迷路データを+1する
C2D5 10BF	DJNZ MALP2	← 仮想迷路データを+1する
C2D7 13	INC DE	← 仮想迷路データを+1する
C2D8 13	INC DE	← 仮想迷路データを+1する
C2D9 0D	DEC C	← 仮想迷路データを+1する
C2DA 20B8	JR NZ,MALP1	← 仮想迷路データを+1する
C2DC C9	RET	
C2DD	RMEDGE: ;ReMove EDGE	← 仮想迷路から周囲のFFHをとる
C2DD 119CC0	LD DE,IMAZE	← 仮想迷路から周囲のFFHをとる
C2E0 21AFC0	LD HL,IMAZE+IMZX+3	← 仮想迷路から周囲のFFHをとる
C2E3 060C	LD B,12	← 仮想迷路から周囲のFFHをとる
C2E5	RELOOP: ;Remove Edge LOOP	
C2E5 C5	PUSH BC	← 仮想迷路から周囲のFFHをとる
23680 C2E6 011000	LD BC,16	← 仮想迷路から周囲のFFHをとる


```

23690 C2E9 EDB0      LDIR
C2EB C1             POP BC
C2EC 23             INC HL
C2ED 23             INC HL
C2EE 10F5           DJNZ RELOOP
C2F0 C9             RET

;
0000      WALPT: EQU 0 ;WALL Pattern number
;
C2F1      DISPMZ: ;Display Maze
C2F1 219CC0         LD HL,IMAZE
C2F4 010000         LD BC,0000H
C2F7      DMLoop: ;Display Maze LOOP
C2F7 C5             PUSH BC
C2F8 E5             PUSH HL
C2F9 7E             LD A,(HL)
C2FA FEFF           CP 0FFH
C2FC 2808           JR Z,DPWALL
C2FE 3287BF         LD (COLOR),A
C301 211004         LD HL,410H
C304 CD49BF         CALL CLPTXY
C307 1805           JR SEEKNP
C309      DPWALL: ;Display WALL
C309 3E00           LD A,WALPT
C30B CD10BF         CALL DISP
C30E      SEEKNP: ;Seek Next Position
C30E E1             POP HL
C30F C1             POP BC
C310 23             INC HL
C311 79             LD A,C
C312 C604           ADD A,4
C314 4F             LD C,A
C315 FE3D           CP 61
C317 380E           JR C,DMLoop
C319 0E00           LD C,0
C31B 78             LD A,B
C31C C604           ADD A,4
C31E 07             LD B,A
C31F FE2D           CP 45
C321 38D4           JR C,DMLoop
C323 C9             RET

;
24100      ***** List 5-1-T *****
;
D000      TEST: ;TEST
D000 F3             DI
D001 AF             XOR A
D002 D351           OUT (51H),A
D004 3100B6         LD SP,STACK
D007 CD88BF         CALL CLS
D00A CD58C2         CALL MKIMZ
D00D CD8CC2         CALL MKAMZ
D010 CD0DC2         CALL RMEDGE
D013 CDF1C2         CALL DISPMZ
D016 FB             EI
50130 D017 FF       RST 38H

```

3. キー入力…操作性の向上

本章のVIPである迷路については、すでにその全貌が明らかになっています。パターンの移動に必要な座標データや、迷路を画面に表示するルーチンも実際に作ってしまったわけですから、ここから先はつけたしに過ぎないかもしれません。別のいい方をすれば、迷路ゲームを完成させるためにあるようなものです。しかし、1つのゲームを作るにあたって、めんどろな作業が多くなるのもこれからです。特に、現実商品となる作品を作る際には、ここからどの程度ゲームに対して「気配り」をできるかがポイントとなってきます。いわゆる原因不明のバグが出てくるものも、これから先にかけてがほとんどですし、単純に迷路についてわかったからといって、気を抜くことはまだまだできません。

ここで作るゲームは、マシン語勉強用であって商品ではありませんから、すべてに渡って「気配り」をすることはしませんが、基本的な部分においては細かな点にまで気を使っています。では、このような迷路型ゲームにおける基本的な部分とは何かというと、それはキー操作が簡単かどうかということです。どんなにアイデアが良くても、主人公を思い通りに動かせなければ、ゲームがつまらなくなってしまう。つまり、キー操作もゲームのアイデアの重要な要素である、ということになるのです。ただし、キー操作だけすぐれていても、ゲームとしての価値はありませんから、やはりメインのアイデアが最重要であることに

変わりはありません。ですから、ここでのキー操作法が、すべての迷路型ゲームについて最適であるとは断言できませんし、そういうものはまた存在しないはずなのです。そのため、オリジナルのゲームを作る際には、そのゲーム内容に応じたキー操作のアイデアも一緒に考える必要があります。

さて、本題に入る前にこの迷路ゲームで使うパターンのデータを一気に作成してしましましょう（パターンはカラーページ④）。本当は、迷路では方向別に2〜3パターンを用意すると、アニメ的に動きがでて良いのですが、テストということで方向別に作るのはやめて、それぞれ2パターンを交互に表示するだけで妥協しました。もし、プログラム・コンテストに応募しようなどと思っている方は、すべてにおいて安易に妥協などしてはいけません。誰が見ても、妥協した所はすぐにわかるものです。どうせなら、これは作るのが大変だったろうな、と想像されるようにしておくべきです!?

ところで、アニメーションといえば、ウォルト・ディズニーのものがキャラクターといい、動きのなめらかさといい、正に世界の最高峰ですが、彼は決して妥協をしなかったそうです。それどころか、常に新しいものへのチャレンジを試み、かならず前作より優れた作品を生み出していったのです。しかし、その陰に隠れて、意外と知られていないのがアブ・アイワークスの存在です。彼の絵を描く能力、機械を扱う能

力というのは、天才的であり、1日に700枚ものデッサンを描いたという話があるくらいです。ミッキー・マウスの生みの親がディズニーなら、育ての親とも生命をふき込んだ男ともいわれているのが、アブ・アイワークスなのです。ディズニーの夢と希望を持つ雰囲気、PC-8801のゲームで中が見たいと思うのですが、残念ながらまだそのような作品に出会ったことも、また作ろうとしても足元にも及ばないというのが現実です。どこかに、和製アブ・アイワークスはいませんか…。さて、パターン・エディタで作成したパターン・データは、指示通りのアドレスに転送してください。また、数字、文字のデータは前回と同じものですが、アドレスが違っていますので、同様に転送をする必要があります。

パターン・データが整えば、残るはプログラムということになります。が、その前にキー・スキャンから移動方向を決定するまでのアルゴリズムを、正確に把握しておかないと、このプログラムは少々難解です。

それは、移動方向の決定に際し、座標データによる制限が加わっているだけでなく、キー操作を簡単にするというテーマも入っているからです。方向番号は前の節で決めた通りですが、追加として停止状態を方向番号=0としています。また、反対方向以外への方向変更は、4コマ毎にしかできないということ来判断するため、図7のように座標データのある位置を基準(0)として、方向別に移動カウンター(0~3)を定めています。この値は、ワークエリア(MYWORK+1)上に保存しておき、移動するたびに方向により増減されることになります。

さてキー入力に対する操作性アップのためのルールは下に示してあります。迷路ゲームではごく当たり前のものです。

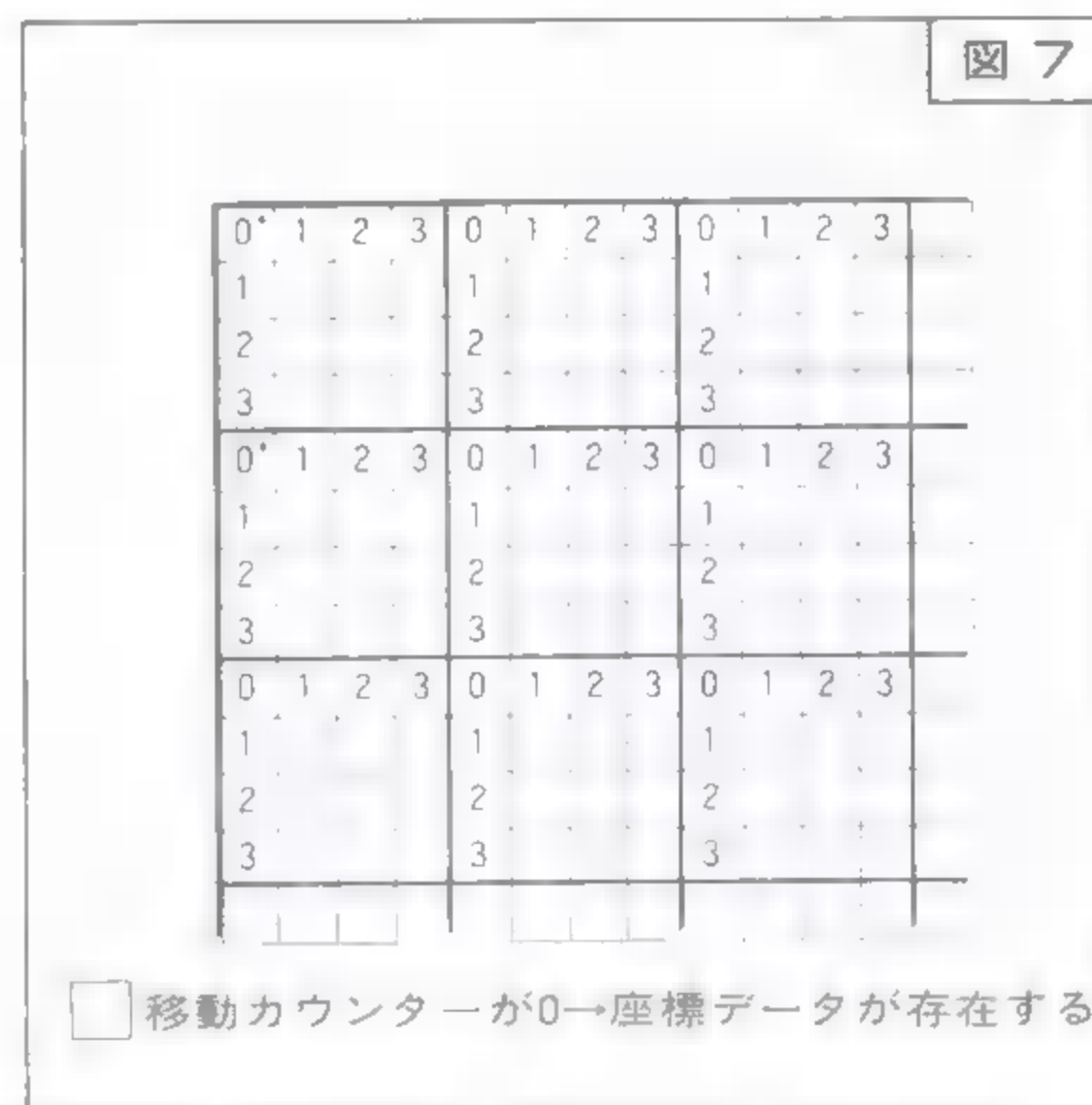
これらのキー入力と移動との関係を、理解した上でList 5-2を見てみることにしましょう。

ここでは、迷路内の移動と同時に、得点の計算および表示も行なっているため、そのルーチンも入っていますが、これらは前

1. 反対方向を示すキー(4と6, 2と8)が、同時に押されている場合は、両方共に押されていないものとする。したがって、3つのキーが同時に押された時は、残る1方向のキーのみが押されていることになる。
2. 反対方向以外の2つのキー(2と6, 6と8, 8と4, 4と2)が、同時に押されている場合は、現在の進行方向でない方向に優先権がある。これにより、カギ形状の道を簡単に進むことができる。
3. パターンが停止できるのは、移動カウンターが0になっている場所だけとし、キーが押されていなくても、移動カウンターが0になるまでは同じ方向に進む。これは1コマだけ位置がズレているために曲がれない、というようなキー操作性の悪さがでないようにするためである。
4. 方向変更は、移動カウンターが0の所では座標データにより、それ以外の地点では反対方向へのみ自由とする。

回のシューティング・ゲームと同じものです。得点は道路を塗るたびに、パレット番号に相当するだけのアップがあります。また、ボーナスとして、全部の道路が同一色になった時(白は除く)には、そのパレットコード×1000点が加算されます。ただしボーナス点の判定には道路の数(ブロック数)が、ワークエリア(PAINWK)に入っていないければなりません。ここでは道路の数を数えるルーチンはまだ作ってありません。そのため、テストを実行してもボーナス点は正しく加算されません。同様に、スコアについても初期値(0点)を入れていませんから、初期の得点は不定です。これらの得点に関しては、プログラムの前回のものとは変わりありません。

このプログラムで中心となるのは、やはり主人公の移動(MYMOVE)についてのルーチンで中でも方向を決定している部分(KEYCHK)がキー・ポイントです。方向が決まれば、パターンの下にある道路の色を



仮想画面(IMAZE)から取り出し、方向別に消去してパターンを移動させればいいのです。移動カウンターの増減や、得点の加算なども方向が決定してからのことになります。そこで、方向決定までのプログラムの手順ですが、わかりやすく書くと次のようになります。

1. 押されたキー(2,4,6,8)を調べる。データは反転して、押されたビットが1になるようにしておく。…押されたキーの値
2. 反対方向のキー(2と8, 4と6)が、同時に押されていれば、その方向のビットを両方共0にして、押されていないものとする。…押されたと判定されたキーの値
3. 移動カウンターが0でない場合、反対方向のキーが押されていれば、その方向に決定し、それ以外はすべて現在方向のままにする。
4. 移動カウンターが0の場合、座標データと「押されたと判定されたキーの値」とのANDを取る。…移動可能と判定されたキーの値
5. 「移動可能と判定されたキーの値」が0でなく、現在方向が01H(上)が04H(下)の時は、左右への方向から優先的に決定する。
6. 「移動可能と判定されたキーの値」が0でなく、現在方向が10H(左)か40H(右)の時は、上下への方向から優先的に決定する。
7. 「移動可能と判定されたキーの値」が0の場合は、停止すなわち方向番号=0とする。

なお、この方向決定のプログラム中で、初めて裏レジスタが使用されています。ここでは、アキュムレータだけについてですが、EXX 命令を使えば BC, DE, HL についての使用が可能です。裏レジスタとは、能力的には表レジスタ(今までのレジスタ)と全く同等で、うまく利用するとレジスタを倍に使うことができる便利なものです。しかし、両方を同じ次元で使うことはできないので、このように利用するたびに交換して使われなければなりません。また、アキュムレータは独立して交換するので、表/裏のペアレジスタ共通の変数に使えることになります。裏レジスタは、一見すると PUSH, POP 命令と同じ感覚で使用できそうですが、2 度繰り返すと元に戻るということを、

常に頭に入れておかないと、どちらのレジスタを使っているのかわからなくなってしまい、大変危険です。そのため、裏レジスタについては、どうしてもレジスタが足りないという場合にだけ使用し、この例のように PUSH, POP 命令でも問題がない場合には、使用しないようにしましょう。

では、テストを実行してプログラムの動作を確認してください。ウェイトを入っていないので、動きが速いかもしれませんが、キーの操作性は合格点のはずです。ゲームのアイディアに対する点数は、…これは、好みの問題(?)ですから、あなた自身の判断で採点をお願いいたします。といっても、またゲームとして完成してないのですから、無理な話でしたね。先へ進みましょう…。

List 5-2 キー入力と

```

12000          ;***** List 5-2-G *****
          ;
          ;DISPLE: ;DISPlay Letter
BFB4          CALL XYADR
BFB4 CD53C0    CALL SEEKLD
BFB7 CD24C3    CALL SEEKLD
BFB8 D35C      OUT (5CH),A
BFB8 CDD2BF    CALL BOXL
BFBF 2AE0BF    LD HL,(LDADR)
BFC2 D35D      OUT (5DH),A
BFC4 CDD2BF    CALL BOXL
BFC7 2AE0BF    LD HL,(LDADR)
BFCA D35E      OUT (5EH),A
BFCC CDD2BF    CALL BOXL
BFCF D35F      OUT (5FH),A
BFD1 C9        RET

          ;
          ;BOXL: ;BOX of Letter
BFD2          LD (LETSP+1),SP
BFD2 ED73EABF  LD SP,HLEN-2
BFD6 314E00    LD DE,(DISPAD)
BFD9 ED5B47BF  LD BC,8FFH
12200 BFD0 01FF08

```

```

BFE0      LLOOP: ;Letter Loop
BFE0 EDA0      LDI
BFE2 EDA0      LDI
BFE4 EB        EX    DE,HL
BFE5 39        ADD   HL,SP
BFE6 EB        EX    DE,HL
BFE7 10F7      DJNZ  LLOOP
BFE9          LFTSP: ;Letter Stack Pointer
BFE9 310000     LD    SP,0000
BFEC C9        RET

;
BFED          LDADR: ;Letter Data Address
BFED          DS    2

;
;***** List 5-2 N *****
;
BCC0          LBASE: EQU 0BCC0H ;Letter LMSL address
;
C324          SEEKLO: ;Letter Data
C324 87        ADD   A,A
C325 87        ADD   A,A
C326 6F        LD    L,A
C327 2600      LD    H,0
C329 29        ADD   HL,HL
C32A 29        ADD   HL,HL
C32B 11C0BC    LD    DE,LBASE
C32E 19        ADD   HL,DE
C32F 22EDBF    LD    (LDADR),HL
C332 C9        RET

;
C333          MVPAIN: ;Move and Paint
C333 C5        PUSH  BC
C334 FE01      CP    DU
C336 2811      JR    Z,UPAIN
C338 FE04      CP    DD
C33A 2825      JR    Z,DPAIN
C33C FE10      CP    DL
C33E 2815      JR    Z,LPAIN

;
C340          RPAIN: ;Right move PAINT
C340 211001    LD    HL,110H
C343 CD49BF    CALL  CLPTXY
C346 C1        POP   BC
C347 0C        INC   C
C348 C9        RET

;
C349          UPAIN: ;Up move PAINT
C349 04        INC   B
C34A 04        INC   B
C34B 04        INC   B
C34C 210404    LD    HL,404H
C34F CD49BF    CALL  CLPTXY
C352 C1        POP   BC
C353 05        DEC   B
C354 C9        RET

;
C355          LPAIN: ;Left move PAINT
C355 0C        INC   C
C356 0C        INC   C
C357 0C        INC   C

```

移動方向別消去および次座標計算
 =01 (150)なら UPAIN
 消去をする
 消去をする

5460	C358 211001	LD HL,110H	消去のサイ
	C35B CD49BF	CALL CLPTXY	からHから18から3
	C35E C1	POP BC	
	C35F 0D	DEC C	1. 次座標
	C360 C9	RET	
	C361	;DPAIN: ;Down move PAINT	
	C361 210404	LD HL,404H	消去のサイ
	C364 CD49BF	CALL CLPTXY	消去をする
	C367 C1	POP BC	
	C368 04	INC B	
	C369 C9	RET	
	0042	;SCLOC: EQU 0042H ;Score Location	
	C36A	;DISPSC: ;DISPlay Score	
	C36A 014200	LD BC,SCLOC	
	C36D 219EC3	LD HL,SCOREL	
	C370 7B	LD A,E	
	C371 86	ADD A,(HL)	
	C372 27	DAA	
	C373 77	LD (HL),A	
	C374 2B	DEC HL	
	C375 7A	LD A,D	
	C376 8E	ADC A,(HL)	
	C377 27	DAA	
	C378 77	LD (HL),A	
	C379 2B	DEC HL	
	C37A 3E00	LD A,0	
	C37C 8E	ADC A,(HL)	
	C37D 27	DAA	
	C37E 77	LD (HL),A	
	C37F CD87C3	CALL SCOREP	
	C382 23	INC HL	
	C383 CD87C3	CALL SCOREP	現在の
	C386 23	INC HL	加算(1)から
	C387	;SCOREP: ;SCORE Print	
	C387 7E	LD A,(HL)	
	C388 07	RLCA	
	C389 07	RLCA	
	C38A 07	RLCA	
	C38B 07	RLCA	
	C38C CD90C3	CALL PRINTF	
	C38F 7E	LD A,(HL)	
	C390	;PRINTF: ;PRINT figure	
	C390 E60F	AND 0FH	
	C392 C5	PUSH BC	
	C393 E5	PUSH HL	
	C394 CD84BF	CALL DISPLE	
	C397 E1	POP HL	
	C398 C1	POP BC	
	C399 0C	INC C	
	C39A 0C	INC C	
	C39B C9	RET	
	C39C	;SCORE2: ;SCORE 2	
	C39C	DS 1	
	C39D	;SCORE1: ;SCORE 1	
	C39D	DS 1	

C39E	SCOREL: ;SCORE Low	
C39E	DS 1	
C39F	PAINWK: ;PAINT Work area	
C39F	DS 7	
C3A6	SPACE: ;SPACE key data	
C3A6	DS 1	
C3A7	MYWORK: ;MY WORK area	---主人公のワークエリア
C3A7	DS 1	移動方向
C3A8	DS 1	移動方向
C3A9	DS 2	移動
C3AB	DS 1	移動
0001	MYPT: EQU 1 ;MY Pattern number	
C3AC	MYMOVE: ;MY MOVE	移動方向決定 () = MYWORK (移動方向) となっている
C3AC CD65C4	CALL KEYCHK	
C3AF 7E	LD A, (HL)	移動方向
C3B0 B7	OR A	
C3B1 2860	JR Z, MYDISP	MYDISP
C3B3 23	INC HL	移動方向
C3B4 56	LD D, (HL)	
C3B5 23	INC HL	
C3B6 4E	LD C, (HL)	
C3B7 23	INC HL	
C3B8 46	LD B, (HL)	
C3B9 F5	PUSH AF	
C3BA CD31C4	CALL GETCOL	移動方向の色を () に入れる
C3BD F1	POP AF	
C3BE F5	PUSH AF	
C3BF CD23C4	CALL CCHAN	移動方向の色を方向別に A に決める
C3C2 21A8C3	LD HL, MYWORK+1	移動方向の色・アドレス
C3C5 86	ADD A, (HL)	
C3C6 E603	AND 3	移動方向
C3C8 77	LD (HL), A	
C3C9 F1	POP AF	移動方向の色を取り出す・移動方向
C3CA ED4BA9C3	LD BC, (MYWORK+2)	移動方向の色
C3CE CD33C3	CALL MVPAIN	移動方向の色・BC は次座標になる
C3D1 ED43A9C3	LD (MYWORK+2), BC	移動方向の色
C3D5 3AA8C3	LD A, (MYWORK+1)	移動方向の色
C3D8 B7	OR A	
C3D9 203B	JR NZ, MYDP2	移動方向
C3DB 3AA6C3	LD A, (SPACE)	
C3DE E640	AND 40H	移動方向の色が押されていれば MYDISP へ
C3E0 2831	JR Z, MYDISP	
C3E2 CD52C4	CALL BCTOIM	移動方向の色・仮想的迷路アドレスをHLに決める
C3E5 7E	LD A, (HL)	移動方向の色(ブロック)の色
C3E6 FE07	CP 7	
C3E8 2829	JR Z, MYDISP	移動方向の色・MYDISP 色の変更なし
C3EA 3C	INC A	
C3EB 77	LD (HL), A	移動方向の色・番号を+1する
C3EC 5F	LD E, A	
C3ED 1600	LD D, 0	
C3EF F5	PUSH AF	移動方向の色・スコアに合わせてスコアをアップする
C3F0 CD6AC3	CALL DISPSC	
C3F3 F1	POP AF	
C3F4 4F	LD C, A	移動方向の色・スコアの色のワークエリアへイ
C3F5 0600	LD B, 0	移動方向の色・スコアの色のワークエリアへイ
C3F7 219EC3	LD HL, PAINWK-1	移動方向の色・スコアの色のワークエリアへイ
C3FA 09	ADD HL, BC	移動方向の色・スコアの色のワークエリアへイ
C3FB 35	DEC (HL)	移動方向の色・スコアの色のワークエリアへイ
C3FC 2015	JR NZ, MYDISP	移動方向の色・スコアの色のワークエリアへイ

26680	C3FE FE07	CP	7	
	C400 2811	JR	Z,MYDISP	A=7ならMYDISPへ…全部の道が白くな…時
	C402 23	INC	HL	
	C403 7E	LD	A,(HL)	…メモリを…残り残し散
	C404	BONSch:	;BONUS Check	
	C404 FE00	CP	0	……………
	C406 200B	JR	NZ,MYDISP	……………
	C408 79	LD	A,C	……………
	C409 87	ADD	A,A	
	C40A 87	ADD	A,A	
	C40B 87	ADD	A,A	
	C40C 87	ADD	A,A	
	C40D 57	LD	D,A	
	C40E 1E00	LD	E,0	
	C410 CD6AC3	CALL	DISPSC	
	C413	MYDISP:	;MY DISPlay	
	C413 3AA8C3	LD	A,(MYWORK+1)	
	C416	MYDP2:	;MY DisPlay	
	C416 E601	AND	1	
	C418 0E01	LD	C,MYPT	
	C41A 81	ADD	A,C	
	C41B ED4BA9C3	LD	BC,(MYWORK+2)	
	C41F CD10BF	CALL	DISP	
	C422 C9	RET		
	C423	;CHAN:	;Counter CHANge	…方向別にカウンタの増減値を求める
	C423 FE01	CP	0H	
	C425 2807	JR	Z,SUBC	
	C427 FE10	CP	0L	
	C429 2803	JR	Z,SUBC	
	C42B 3E01	LD	A,1	……………
	C42D C9	RET		
	C42E	SUBC:	;SUBtract Counter	……下または左へ…
	C42E 3EFF	LD	A,-1	
	C430 C9	RET		
	C431	;GETCOL:	;GET COLOR number	……………OR……………
	C431 5F	LD	E,A	
	C432 CD52C4	CALL	BCTOIM	……………
	C435 7A	LD	A,D	
	C436 B7	OR	A	……………
	C437 2814	JR	Z,GETCO2	
	C439 7B	LD	A,E	
	C43A FE04	CP	0D	……………
	C43C 280F	JR	Z,GETCO2	
	C43E FE40	CP	0R	……………
	C440 280B	JR	Z,GETCO2	
	C442 010100	LD	BC,1	……………
	C445 FE10	CP	0L	……………
	C447 2803	JR	Z,GETCO1	……………GETCO1
	C449 011000	LD	BC,IMZX	……………
	C44C	GETCO1:	;GET Color 1	
	C44C 09	ADD	HL,BC	
	C44D	GETCO2:	;GET Color 2	
	C44D 7E	LD	A,(HL)	……………
	C44E 3287BF	LD	(COLOR),A	
	C451 C9	RET		
	C452	;BCTOIM:	;BC TO Image Maze	(C,B)の属するブロックについて仮想迷路 アドレスをHLに求める
	C452 CB39	SRL	C	
27280	C454 CB39	SRL	C	

C456	78	LD	A,B
C457	87	ADD	A,A
C458	87	ADD	A,A
C459	E6F0	AND	0F0H
C45B	6F	LD	L,A
C45C	2600	LD	H,0
C45E	44	LD	B,H
C45F	09	ADD	HL,BC
C460	019CC0	LD	BC,IMAZE
C463	09	ADD	HL,BC
C464	C9	RET	
;			
C465		KEYCHK:	
C465	DB00	IN	A,(0)
C467	F6AB	OR	0ABH
C469	47	LD	B,A
C46A	DB01	IN	A,(1)
C46C	A0	AND	E
C46D	2F	CPL	
C46E	47	LD	B,A
C46F	4F	LD	C,A
C470	0F	RRCA	
C471	3008	JR	NC,KCK1
C473	0F	RRCA	
C474	0F	RRCA	
C475	3004	JR	NC,KCK1
C477	3EF0	LD	A,0F0H
C479	A0	AND	B
C47A	47	LD	B,A
C47B		KCK1: ;key check	
C47B	79	LD	A,C
C47C	07	RLCA	
C47D	07	RLCA	
C47E	3008	JR	NC,KCK2
C480	07	RLCA	
C481	07	RLCA	
C482	3004	JR	NC,KCK2
C484	3E0F	LD	A,0FH
C486	A0	AND	B
C487	47	LD	B,A
C488		KCK2: ;key check	
C488	21A8C3	LD	HL,MYWORK+1
C48B	7E	LD	A,(HL)
C48C	B7	OR	A
C48D	2829	JR	Z,JUST
C48F	2B	DEC	HL
C490	7E	LD	A,(HL)
C491	A0	AND	B
C492	C0	RET	NZ
C493	7E	LD	A,(HL)
C494	FE01	CP	DU
C496	280E	JR	Z,DNCK
C498	FE04	CP	DD
C49A	2810	JR	Z,UPCK
C49C	FE10	CP	DL
C49E	2812	JR	Z,RICK
C4A0		LECK: ;Left check	
C4A0	CB60	BIT	4,B
C4A2	C8	RET	Z
C4A3	3610	LD	(HL),DL
C4A5	C9	RET	

C4A6	DNCK:	;Down check	
C4A6 CB50		BIT 2,B	
C4A8 C8		RET Z	
C4A9 3604		LD (HL),DO	
C4AB C9		RET	
C4AC	UPCK:	;UP check	
C4AC CB40		BIT 0,B	
C4AE C8		RET Z	
C4AF 3601		LD (HL),DU	
C4B1 C9		RET	
C4B2	RICK:	;Right check	
C4B2 CB70		BIT 6,B	
C4B4 C8		RET Z	
C4B5 3640		LD (HL),DR	
C4B7 C9		RET	
C4B8	JUST:	;JUST counter	
C4B8 C5		PUSH BC	
C4B9 23		INC HL	
C4BA 4E		LD C,(HL)	
C4BB 23		INC HL	
C4BC 46		LD B,(HL)	
C4BD CDF6C4		CALL GETARR	
C4C0 2B		DEC HL	
C4C1 2B		DEC HL	
C4C2 2B		DEC HL	
C4C3 C1		POP BC	
C4C4 4F		LD C,A	
C4C5 78		LD A,B	
C4C6 A1		AND C	
C4C7 282B		JR Z,NOMAT	
C4C9 08		EX AF,AF	
C4CA 7E		LD A,(HL)	
C4CB E605		AND 5	
C4CD 2813		JR Z,JUST1	
C4CF 08		EX AF,AF	
C4D0 3640		LD (HL),DR	
C4D2 07		RLCA	
C4D3 07		RLCA	
C4D4 D8		RET C	
C4D5 3610		LD (HL),DL	
C4D7 07		RLCA	
C4D8 07		RLCA	
C4D9 D8		RET C	
C4DA 3604		LD (HL),DO	
C4DC 07		RLCA	
C4DD 07		RLCA	
C4DE D8		RET C	
C4DF 3601		LD (HL),DU	
C4E1 C9		RET	
C4E2	JUST1:	;JUST 1	
C4E2 08		EX AF,AF	
C4E3 3601		LD (HL),DU	
C4E5 0F		RRCA	
C4E6 D8		RET C	
C4E7 3604		LD (HL),DO	
C4E9 0F		RRCA	
C4EA 0F		RRCA	
C4EB D8		RET C	
28490 C4EC 3610		LD (HL),DL	

C4EE 0F	RRCA		
C4EF 0F	RRCA		これはリター
C4F0 08	RET C		
C4F1 3640	LD (HL),DR		
C4F3 C9	RET		
C4F4	NOMAT: ;NO MATCH		
C4F4 77	LD (HL),A		
C4F5 C9	RET		
C4F6	;		
C4F6 E5	GETARR: ;GET ARROW data		——Aに座標データを入れる
C4F7 CB39	PUSH HL		
C4F9 CB39	SRL C		
C4FB 78	SRL C		
C4FC 87	LD A,B		
C4FD 87	ADD A,A		
C4FE 6F	ADD A,A		
C4FF 2600	LD L,A		
C501 44	LD H,0		
C502 09	LD B,H		
C503 0198C1	ADD HL,BC		
C506 09	LD BC,AMAZE		
C507 7E	ADD HL,BC		
C508 E1	LD A,(HL)		
C509 C9	POP HL		
	RET		

***** List 5-2-1 *****

D000	TEST:	; TEST	
D000 F3		DI	
D001 AF		XOR A	
D002 D351		OUT (51H),A	20000000
D004 3100B6		LD SP,STACK	
D007 CD88BF		CALL CLS	00000000
D00A CD58C2		CALL MKIMZ	00000000
D00D CD8CC2		CALL MKAMZ	00000000
D010 CDDDC2		CALL RMEDGE	00000000
D013 CDF1C2		CALL DISPMZ	00000000
D016 AF		XOR A	
D017 21A7C3		LD HL,MYWORK	
D01A 77		LD (HL),A	主人公
D01B 23		INC HL	移動
D01C 77		LD (HL),A	移動
D01D 23		INC HL	移動
D01E 77		LD (HL),A	移動
D01F 23		INC HL	
D020 77		LD (HL),A	
D021	; TLOOP:	; test LOOP	
D021 DB09		IN A,(9)	A ← 入力ポートの9番の値
D023 32A6C3		LD (SPACE),A	SPACE ←
D026 0F		RRCA	
D027 3005		JR NC,TEND	押されていれは TEND →
D029 CDACC3		CALL MYMOVE	主人公の移動
D02C 18F3		JR TLOOP	TLOOP →
D02E	TEND:	; Test END	
D02E FB		EI	
D02F FF		RST 38H	押されたキーの ビ ビ ビ

50310

押されたキーの値

ビット	ビット	ビット	ビット
ト	ト	ト	ト
6	4	2	0
6	4	2	8

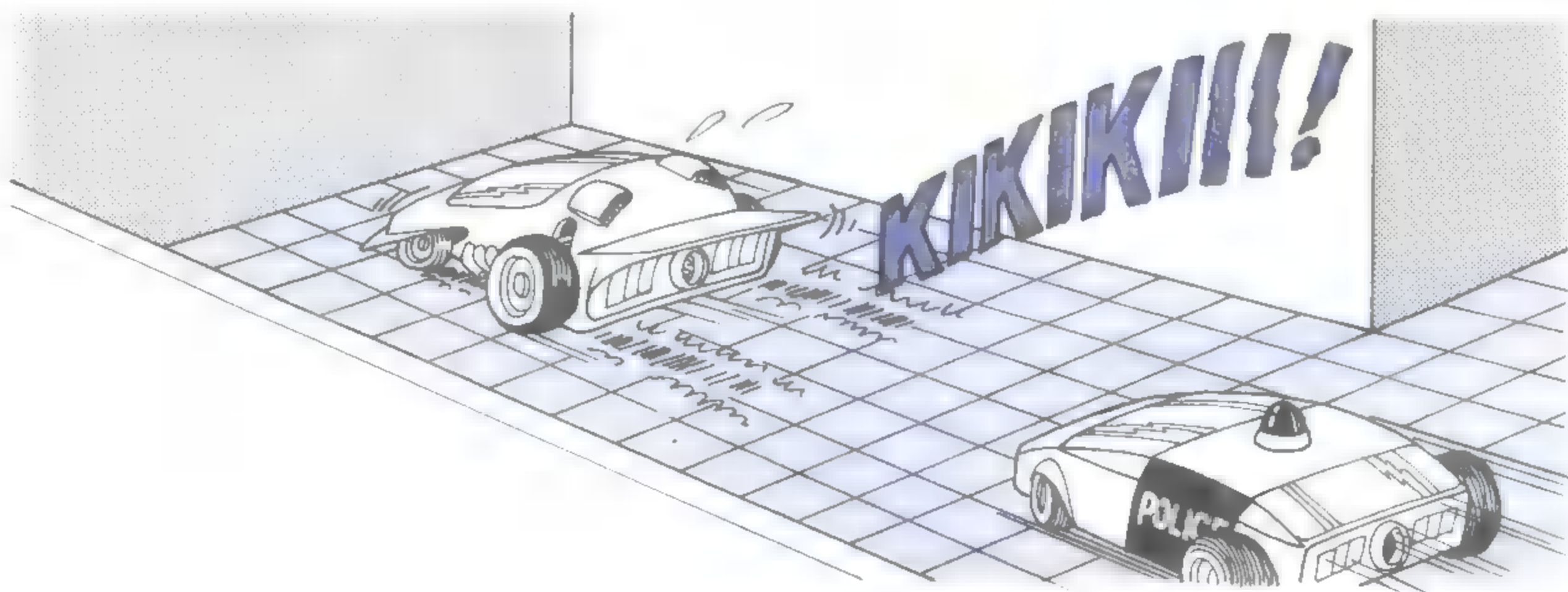
4. 追跡…サァー, 追いかけてよう!

テレビ・ドラマや映画を見ていると、かならず主人公を悩ますイジワルな人物が登場しますが、冷静に考えれば彼等の存在により物語が進行しているのです。さらに何も事件の起きない平和は場面ばかりでは、見ても面白くも何ともないわけです。ウマイ役者であればあるほど、見るからに憎たらしい演技をし、まるでそれが本人の性格であるような錯覚を起こさせます。そのため、悪役というのはいつも悪役になるケースが多く、また見る方もその方が安心して見られるということになります。これとは逆に、悪いことはできないというイメージが定着すると、どんなに演技がうまくても悪役は似合わなくなってしまうから不思議なものです。

ゲームの世界は、映画などに比べるとはるかに《小さな世界》ですから、このように見ただけでそのイメージが強烈に沸いてくる、ということとはあまり考えられません。

そのため、1つのゲームの中でプレイヤーに「憎たらしいヤツだ」とか、「バカなヤツだ」と思われるように、わかりやすい性格をつけてやる必要があります。主人公には、キー入力による制限で簡単に性格をつけてやれますが、悪役となる敵に対しては知恵を授けてやらなければなりません。

何だかむずかしいテーマのようになってきましたが、迷路型ゲームにおいてどのような時に敵が賢く見えるかを考えれば、答えは簡単明瞭です。それは、敵が主人公を追いかける、あるいは遠くから近づいて来る、ということが出来るかどうかです。これを知能と呼ぶには、あまりにもアツカマシイかもしれませんが、前回のゲームではデータによって移動方向を決めていただけですから、追跡をするということは、偉大なる知恵がついたといえるわけです。もし、これが主人公の次の動きを想定して動いてくれるのであれば、本物の人口知能



になれるのですが、実際には主人公のワークエリアにある座標を見て動くだけです。から、悲しいかなイカサマの知能でしかありません。

イカサマの知能を、いかにして本物らし

く見せるか、それがここでのテクニックということであり、また敵に与える性格なのです。そこで、3種類の敵に対して、次のような性格をつけて、プレイヤーをだまそうとしてみます。

敵のタイプ番号=0(パターン番号では3,4)…フラフラ

敵のタイプ番号=1(パターン番号では5,6)…追いかける

敵のタイプ番号=2(パターン番号では7,8)…気まぐれ

フラフラは、乱数との組み合わせでランダムに動くだけです。追いかけるは、ここでの追跡ルーチンにしたがい主人公を追いかけます。気まぐれは、16ブロック移動するたびにフラフラと追いかけるの動きを交互に繰り返していきます。ランダムに動くという

ことは、座標データから1方向を選べばいいのですから、乱数を利用すれば簡単にできそうです。ということは、性格の違う3の敵がいるといっても、重要なのは追跡ルーチンを確立することだけになるわけです。そこで、まずはどのように追いかける

図 8

追跡のルール

1. 座標データから、行ける方向数(矢印の数)を数える
2. 数えた値が1の時は、座標データ通りの方向(袋小路)
3. 数えた値が2の時は、反対方向でない方向(一本道)
- 以上は、フラフラと共通 —
4. 座標データから、反対方向を除く(Uターンの禁止)
5. 主人公の座標=(L, H)、敵の座標=(C, B)とする
6. 方向の決定……反対方向を除いた座標データに、①②③の順に優先権をつけ、移動方向を決定する

主人公と敵との位置	$B \geq H, C \geq L$	$B \geq H, C < L$	$B < H, C \geq L$	$B < H, C < L$
Y軸の差 > X軸の差 $IB - HI$ $IC - LI$	1 2 3 ↑ ← 残り	1 2 3 ↑ → 残り	1 2 3 ↓ ← 残り	1 2 3 ↓ → 残り
Y軸の差 ≤ X軸の差 $IB - HI$ $IC - LI$	1 2 3 ← ↑ 残り	1 2 3 → ↑ 残り	1 2 3 ← ↓ 残り	1 2 3 → ↓ 残り

注：残りの中での優先権は定めず、乱数との組み合わせで決定する

のか、追跡の方針を決定しなければなりません。追跡とは、すなわち移動方向を一定の条件に基いて決めることですが、このゲームにおいては、その前提条件として「Uターンおよび停止はしない」ということにしてあります。ただし、迷路によっては袋小路があることも考えられるので、その際はUターンをすることにします。また、方向の変更があるのは、当然のことですが移動カウンター=0地点(座標データのある場所)だけになります。

この図8の中で、優先方向の最後に「残り」というのがあります。これは基本的には行きたくない方向なので、たとえ2方向が残っていても優先権はつけず乱数によってどちらかを選択するようにしているのです。ここでも、追跡にある程度の自由性を持たせているわけです。

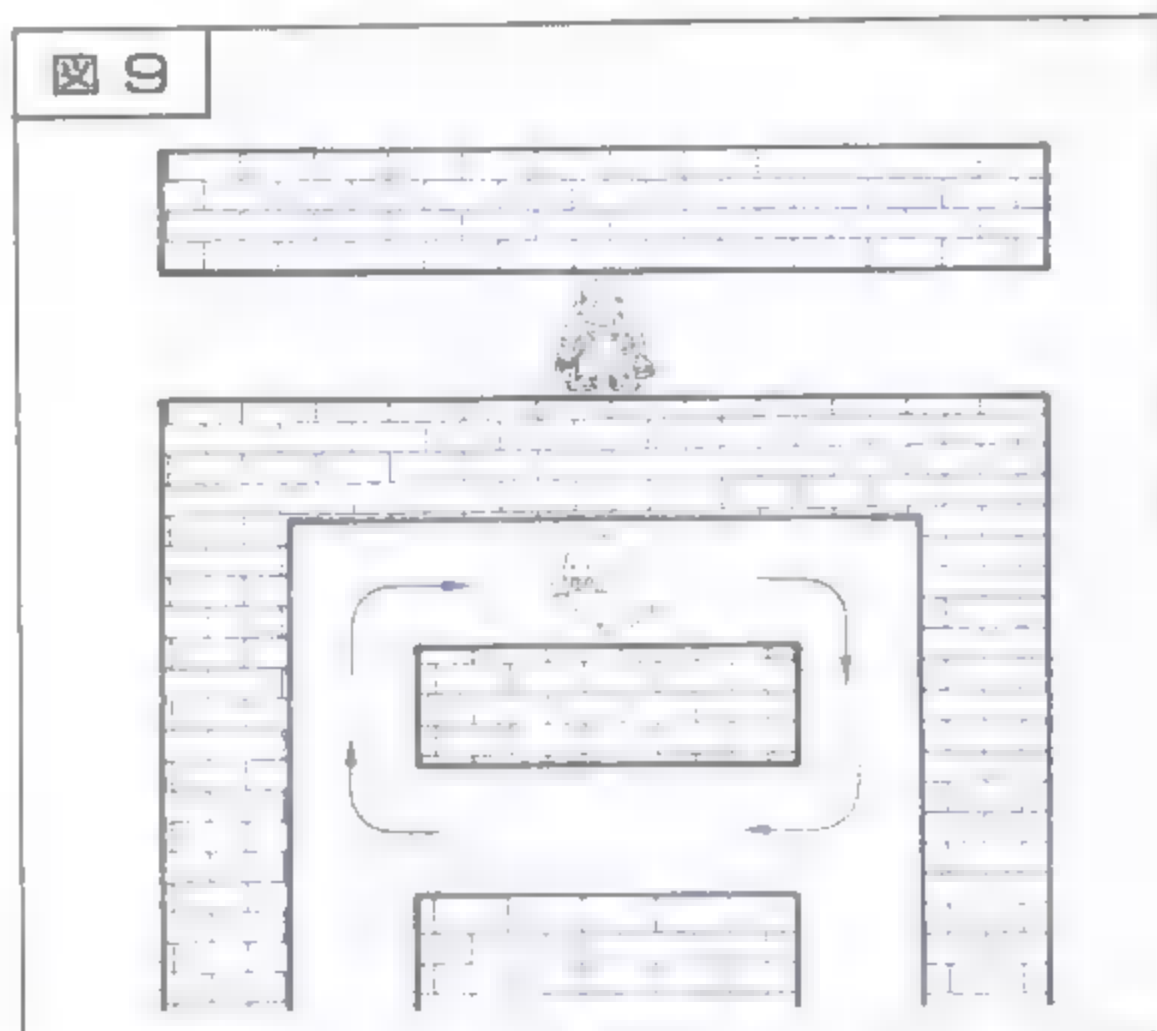
この追跡のルールは、追跡としては最も基本的なパターンなのですが、迷路の形によっては図9のように同じ所をグルグル回ってしまうという欠点があります。

これは、仕方がないといってしまうまでもありますが、知能的な動きからはあまりにもかけ離れています。一々簡単な解決方法は、このような動きが出ないような迷路にすることなのですが、最近は迷路にコンストラクション・セットをつける場合も多くなっています。そのような時に、知恵を与えた敵がこの程度の状態から抜け出せないのでは、すでにプレイヤーとの勝負に負けていることになり、作者として非常にクヤシイ思いがします。そこで、追跡ルーチンの最初に乱数を取り、少ない確率ではあるがランダムに動くルーチンに

も分岐するようにするのです。こうすれば、何回かグルグル回れば、かならず抜け出るような動きをしてくれるので、いかにも敵が自分の頭で考えているように見えてきます。答えがわかると「ナーンだ!!」ということになってしまいますが、要はいかにして《…らしく見せる》かですから、簡単なほどいいのです。ゲームとは、つきつめればキツネとタヌキのたまし合い、いやプレイヤーと作者との知恵比べみたいなものですからね。簡単なテクニックでプレイヤーをダメせれば、作る側の勝ちということです。

追跡の内容が理解できれば、このプログラムも意外と短く感じるかもしれません。結局、キー入力による方向決定にしても、この追跡ルーチンにしても、長く見えるのは方向あるいは位置別に、同じようなプログラムを作らなければならないためで、1つ1つはそれほどのことではないのです。案外、このあたりがマシン語をむずかしく思わせていた理由だったのかもしれません。

さて、このテスト・プログラムを見ると、まず最初にRレジスタの値を乱数の初期値として、ワークエリアに取り入れています。これは、初めて出てきたレジスタですが、このR(リフレッシュ)レジスタというのは、7ビットしかなくダイナミックRAMのリフレッシュ・カウンターとして使われています。RAMとは、一般に電源を切らない限り、その内容が保存されるとだけ示されていますが、RAMにはスタティック型とダイナミック型があり、ダイナミックRAMは極めて短時間で、メモリ内容が消えてしまうのです。そのため、内容が消える前に再びメモリーに書き込む、という作



業をしなければなりません。この書き込みのことをリフレッシュといい、そのためのカウンターになっているのがRレジスタなのです。もちろん、これは内部で自動的にこなわれています。64KRAMの場合、リフレッシュは一度に、512バイトずつされるので、Rレジスタは7ビットで全メモリー空間をカバーできるわけです。ですか

ら、Rレジスタの値は常に変化しており、この値そのものを乱数として使うこともできる位です。ただ、連続して読んだ場合にはあまり変化がないので(+1ずつ増加していく)、ここでは初期値としてだけ用いています。一方、スタティックRAMはこのようなめんどろなことをする必要はないのですが、ダイナミックRAMに比べ価格がはるかに高いため、パソコンクラスではあまり利用されていないのが実情です。

では、テストを実行して追跡のでき具合を見てみましょう。どこに逃げても、アッという間に追いかけてくるはずですが、このテストでは、敵の追いかけてを1種だけにしていますが、これを決めているのは敵のタイプ番号ですから、(IX+0)の値を0または2にすることにより、それぞれフラフラと気まぐれに変化します。性格によって、動きがどのように違うか、試しに確認してみてください。

List 5-3 敵の行動上追跡

```

;***** List 5-3-N *****
;
RND:  ;Random figure
C50A          PUSH HL
C50A E5       PUSH DE
C50B D5       LD HL,(RNDWOK)
C50C 2A1FC5   LD D,H
C50F 54       LD E,L
C510 5D       ADD HL,HL
C511 29       ADD HL,HL
C512 29       ADD HL,DE
C513 19       LD DE,3211H
C514 111132   ADD HL,DE
C517 19       LD (RNDWOK),HL
C518 221FC5   LD A,H
C518 7C       POP DE
C51C D1       POP HL
C51D E1       RET
C51E C9

```

```

C51F      RNDWOK:
C51F      DS      2
C521      EMWORK:
C521      DS      15

;
EMMOVE:
C530      LD      A,(IX+2)
C530 DD7E02      OR      A
C533 B7      CALL   Z,EMJUST
C534 CC72C5      LD      C,(IX+3)
C537 DD4E03      LD      B,(IX+4)
C53A DD4604      LD      D,(IX+2)
C53D DD5602      LD      A,(IX+1)
C540 DD7E01      PUSH   BC
C543 C5      CALL   GETCOL
C544 CD31C4      LD      A,(IX+1)
C547 DD7E01      LD      B,A
C54A 47      CALL   CCHAN
C54B CD23C4      ADD     A,(IX+2)
C54E DD8602      AND     3
C551 E603      LD      (IX+2),A
C553 DD7702      LD      A,B
C556 78      POP     BC
C557 C1      CALL   MVPAIN
C558 CD33C3      LD      (IX+3),C
C55B DD7103      LD      (IX+4),B
C55E DD7004      LD      A,(IX+0)
C561 DD7E00      ADD     A,A
C564 87      ADD     A,3
C565 C603      LD      E,A
C567 5F      LD      A,(IX+2)
C568 DD7E02      AND     1
C56B E601      ADD     A,E
C56D 83      CALL   DISF
C56E CD10BF      RET
C571 C9

;
EMJUST:
C572      LD      C,(IX+3)
C572 DD4E03      LD      B,(IX+4)
C575 DD4604      CALL   GETARR
C578 CDF6C4      LD      B,7
C57B 0607      LD      E,A
C57D 5F      LD      D,A
C57E 57      XOR     A
C57F AF      LD      C,A
C580 4F      ;
EJLOOP:
C581      RR      D
C581 CB1A      ADC     A,C
C583 89      DJNZ   EJLOOP
C584 10FB      CP      3
C586 FE03      JP      C,TAR12
C588 DAB5C5      LD      A,(IX+0)
C58B DD7E00      OR      A
C58E B7      JR      Z,FREEM
C58F 2842      DEC     A
C591 3D      JR      Z,BCHASE
C592 2857

;
CANOF:
C594      LD      HL,CFWORK
C594 21B4C5      DEC     (HL)
C597 35
C598      CKCF:
C598 C2D3C5      JP      NZ,FREEM

```

C59B 3610	LD (HL),16
C59D 3A99C5	LD A,(CKCF+1)
C5A0 FED3	CP FREEM
C5A2 2808	JR Z,CHANC
C5A4 21D3C5	LD HL,FREEM
C5A7 2299C5	LD (CKCF+1),HL
C5AA 1827	JR FREEM
C5AC	
C5AC 21F2C5	LD HL,CHASE
C5AF 2299C5	LD (CKCF+1),HL
C5B2 183E	JR CHASE
C5B4	
C5B4	DS 1
C5B5	
C5B5 FE01	CP 1
C5B7 2004	JR NZ,TAR2
C5B9 DD7301	LD (IX+1),E
C5BC C9	RET
C5BD	
C5BD CDC4C5	CALL EROPP
C5C0 DD7701	LD (IX+1),A
C5C3 C9	RET
C5C4	
C5C4 DD7E01	LD A,(IX+1)
C5C7 47	LD B,A
C5C8 E650	AND 50H
C5CA 3E05	LD A,5
C5CC 2002	JR NZ,EROPP1
C5CE 3E50	LD A,50H
C5D0	
C5D0 B0	OR B
C5D1 A3	AND E
C5D2 C9	RET
C5D3	
C5D3 CDC4C5	CALL EROPP
C5D6 4F	LD C,A
C5D7	
C5D7 CD0AC5	CALL RND
C5DA A1	AND C
C5DB 28FA	JR Z,SKFD
C5DD 4F	LD C,A
C5DE A0	AND B
C5DF C0	RET NZ
C5E0	
C5E0 CD0AC5	CALL RND
C5E3 A1	AND C
C5E4 EAE0C5	JP PE,SKFD1
C5E7 DD7701	LD (IX+1),A
C5EA C9	RET
C5EB	
C5EB CD0AC5	CALL RND
C5EE E60F	AND 0FH
C5F0 28E1	JR Z,FREEM
C5F2	
C5F2 CDC4C5	CALL EROPP
C5F5 08	EX AF,AF
C5F6 2AA9C3	LD HL,(MYWORK+2)

C5F9	DD4E03	LD	C,(IX+3)
C5FC	DD4604	LD	B,(IX+4)
C5FF	78	LD	A,B
C600	BC	CP	H
C601	3851	JR	C,EPOSU

;EPOSU: ;Enemy POS. Up			
C603			
C603	79	LD	A,C
C604	BD	CP	L
C605	3829	JR	C,EPOSU
;EPOSU: ;Enemy POS. Up			
C607			
C607	78	LD	A,B
C608	94	SUB	H
C609	57	LD	D,A
C60A	79	LD	A,C
C60B	95	SUB	L
C60C	5F	LD	E,A
C60D	08	EX	AF,AF
C60E	6F	LD	L,A
C60F	7B	LD	A,E
C610	BA	CP	D
C611	300F	JR	NC,EPDR1
C613	3E01	LD	A,DU
C615	A5	AND	L
C616	C2A2C6	JP	NZ,OKDIR
C619	3E10	LD	A,DL
C61B	A5	AND	L
C61C	C2A2C6	JP	NZ,OKDIR
C61F	C39EC6	JP	TOSF1
;EPDR1: ;Enemy POS. Down & Right			
C622			
C622	3E10	LD	A,DL
C624	A5	AND	L
C625	C2A2C6	JP	NZ,OKDIR
C628	3E01	LD	A,DU
C62A	A5	AND	L
C62B	2075	JR	NZ,OKDIR
C62D	C39EC6	JP	TOSF1

;EPOSU: ;Enemy POS. Up			
C630			
C630	78	LD	A,B
C631	94	SUB	H
C632	57	LD	D,A
C633	7D	LD	A,L
C634	91	SUB	C
C635	5F	LD	E,A
C636	08	EX	AF,AF
C637	6F	LD	L,A
C638	7B	LD	A,E
C639	BA	CP	D
C63A	300C	JR	NC,EPDL1
C63C	3E01	LD	A,DU
C63E	A5	AND	L
C63F	2061	JR	NZ,OKDIR
C641	3E40	LD	A,DR
C643	A5	AND	L
C644	205C	JR	NZ,OKDIR
C646	1856	JR	TOSF1
;EPDL1: ;Enemy POS. Down & Left			
C648			
C648	3E40	LD	A,DR
C64A	A5	AND	L
C64B	2055	JR	NZ,OKDIR
C64D	3E01	LD	A,DU

```

C64F A5          AND L
C650 2050        JR NZ,OKDIR
C652 184A        JR TOSF1

;
EPOSU:           ;Enemy POSition Up
C654             LD A,C
C654 79          CP L
C655 BD          JR C,EPOSUL
C656 3824        EPOSUR: ;Enemy Up ■ Right
C658             LD A,H
C658 7C          SUB B
C659 90          LD D,A
C65A 57          LD A,C
C65B 79          SUB L
C65C 95          LD E,A
C65D 5F          EX AF,AF
C65E 08          LD L,A
C65F 6F          LD A,E
C660 7B          CP D
C661 BA          JR NC,EPUR1
C662 300C        LD A,DD
C664 3E04        AND L
C666 A5          JR NZ,OKDIR
C667 2039        LD A,DL
C669 3E10        AND L
C66B A5          JR NZ,OKDIR
C66C 2034        JR TOSF1
C66E 182E        EPUR1: ;EPoSUR 1
C670             LD A,DL
C670 3E10        AND L
C672 A5          JR NZ,OKDIR
C673 202D        LD A,DD
C675 3E04        AND L
C677 A5          JR NZ,OKDIR
C678 2028        JR TOSF1
C67A 1822        ;
EPOSUL:          ;Enemy Up ■ Left (B<H,C<Lの場合)
C67C             LD A,H
C67C 7C          SUB B
C67D 90          LD D,A
C67E 57          LD A,L
C67F 7D          SUB C
C680 91          LD E,A
C681 5F          EX AF,AF
C682 08          LD L,A
C683 6F          LD A,E
C684 7B          CP D
C685 BA          JR NC,EPUL1
C686 300C        LD A,DD
C688 3E04        AND L
C68A A5          JR NZ,OKDIR
C68B 2015        LD A,DR
C68D 3E40        AND L
C68F A5          JR NZ,OKDIR
C690 2010        JR TOSF1
C692 180A        EPUL1: ;EPoSUL
C694             LD A,DR
C694 3E40        AND L
C696 A5          JR NZ,OKDIR
C697 2009        LD A,DD
C699 3E04        AND L
C69B A5          JR NZ,OKDIR
C69C 2004

```

```
C69E      TOSF1:      ;      skfd1
C69E 4D      LD      C,L
C69F C3E0C5  JP      SKFD1
C6A2      OKDIR:      ;      DIRecti
C6A2 DD7701  LD      (IX+1),A
C6A5 C9      RET

;
;***** List 5-3-T *****
;
D000      TEST:      ;
D000 F3      DI
D001 AF      XOR     A
D002 D351      OUT    (51H),A
D004 3100B6    LD      SP,STACK
D007 E05F      LD      A,R
D009 321FC5    LD      (RNDWOK),A
D00C CD888F    CALL    CLS
D00F CD58C2    CALL    MKIMZ
D012 CD8CC2    CALL    MKAMZ
D015 CDDDC2    CALL    RMEDGE
D018 CDF1C2    CALL    DISPMZ
D01B AF      XOR     A
D01C 21A7C3    LD      HL,MYWORK
D01F 77      LD      (HL),A
D020 23      INC     HL
D021 77      LD      (HL),A
D022 23      INC     HL
D023 77      LD      (HL),A
D024 23      INC     HL
D025 77      LD      (HL),A
D026 DD2121C5 LD      IX,EMWORK
D02A DD360001 LD      (IX+0),1
D02E DD360110 LD      (IX+1),DL
D032 DD360200 LD      (IX+2),0
D036 DD360300 LD      (IX+3),0
D03A DD36042C LD      (IX+4),44
D03E      TLOOP:      ;      Test LOOP
D03E DB09      IN      A,(9)
D040 32A6C3    LD      (SPACE),A
D043 0F      RRCA
D044 3008      JR      NC,TEND
D046 CD30C5    CALL    EMMOVE
D049 CDACC3    CALL    MYMOVE
D04C 18F0      JR      TLOOP
D04E      TEND:      ;      EN
D04E FB      EI
D04F FF      RST     38H
```

■ワークエリアの内容

(1X+0)	タイプ
(1X+1)	移動方向
(1X+2)	移動カウンター
(1X+3)	X 座標
(1X+4)	Y 座標

敵のタイプ	1
移動方向	左
移動カウンター	0
X 座標	0
Y 座標	44

5. 完成…メッセージや音を付ける

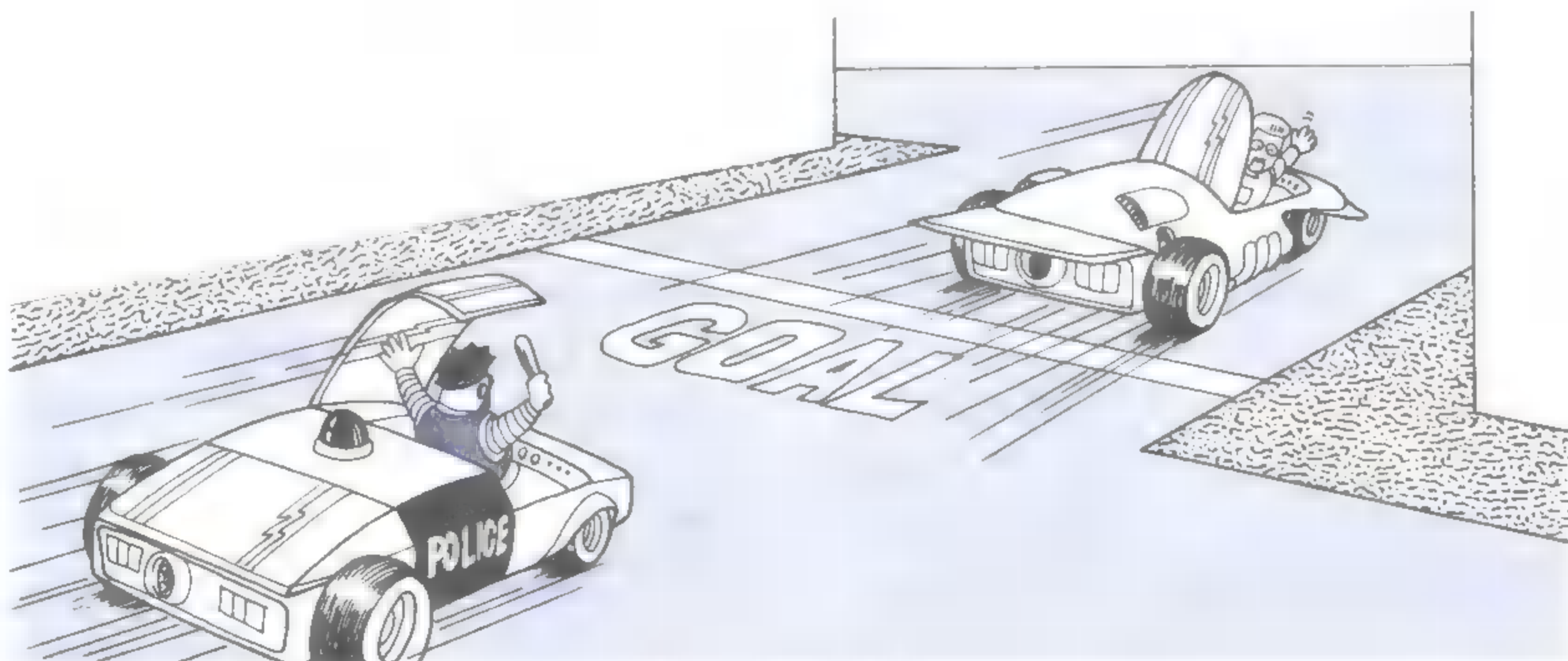
プログラムに限らず、完成直前の状態というのは、誰でも一番胸がワクワクするものです。これは、苦勞の多いものほどうれしさも多く、できることならいつまでも完成直前のままにしておきたい、などという人もいるくらいです。しかし、ゲーム・プログラムのように完成後に楽しみがある場合はそんな悠長な思いは起きませんね。一刻も早く仕上げて、遊んでみたいと思うのが人情というものです。

しかし、長いプログラムを自分で作ると、この完成直前の期間が一番長く感じられ、イヤなものになってしまいます。その原因は、いわずと知れたバグという、コンピュータにとって生まれた時から永遠に続く、運命

共同体があるからです。特に、大きな作品になればなるほど、この時期になって出てくるオカシナ現象に悩まされるようになり、最後には折角自分で遊ぼうと思って作ったゲームなのに、見るのもイヤということになってしまうのです。そういうことが、わかっていながらまた次の作品を作りたくなってくるのは、やはり一種のコンピュータ病なのかもしれません。

本書のプログラムは、そういうことが起きないように、すでに十分に試験走行をしておりますので、安心してください。では最後に次の4つの内容を付加して本章のプログラムを完成しましょう。

1. 文字連続表示ルーチン…前回と同じ
2. 道の数进行を数えるルーチン…ペイントされる道(ブロック)の総数
3. 全敵移動ルーチン…3種類の敵を動かす
4. 衝突判定ルーチン…前回と同じ



たったこれだけですから、説明を必要とするほど複雑なものではなく、プログラムを見れば一目瞭然だと思います。どちらかというところ、ここではテストの部分の量を、前回とは違って商品の一步手前まで高めてあります。

まずノン・グラフィック・ルーチンのウェイトとして、無駄命令ではなく音楽ルーチン(MUSIC)を用いてみました。本当は、完全な音楽を移動用としてつけたかったのですが、プログラムの長さが MF-ASM2 の限界を越えてしまうので、残念ながらただ音を出すだけになってしまいました。そのため、ここでは主人公の移動カウンターを利用し、カウンター番号に合わせてソ・ラ・シ・ドの音を出しているだけですが、音楽専用のカウンターとデータさえ用意すれば、まったく同じ方法で BGM が出せるようになります。簡単なテクニックなのに、PC-8801 で BGM のついたソフトは意外と少ないのです。オリジナル・ゲームを作る時には、ぜひチャレンジしてみてください。また、移動がない時には音が出ないようにしていますが、全体の速度が変化しないようにするには、その分ウェイトをいれなくてやらなければなりません。そこで、同じ音楽ルーチンから休符を利用して、これをウェイトとして使っています。このように、休符は単なるウェイトとしての役目も果たせますから、無駄命令の代わりに利用するとキメの細かなウェイトができます。

さらに、ゲーム完成後にメッセージを入れたり、ゲームが終わってもモニタへ無造作に戻さず、スペースバーで再ゲームができるようにした点などが、前回のものに比



べると進歩しているといえます。しかし、いくら進歩したといっても、これだけでは本物のゲームといえないのは、誰が見ても明らかなことです。このことは、技術的なこと以外に、どこかに致命的な欠陥があるといわざるを得ません。

それは、この程度の長さのプログラムが MF-ASM2 の限界だということに、すべての原因があるのです。それでは、高いお金を出して本格的なアセンブラを買わなければならないのでしょうか。そんな殺生な話はありませんね。何事も工夫とアイディアさえあれば、何とかなるはずです。6 章では、本書最後のゲームとして、市販の商品に負けられないような大作を、この MF-ASM2 で作っていきます。

List 5-4 ペンキ・ボーイの仕上げ

			;***** List 5-4-N *****	
			;MSGPRN: ;MeSsaGe PRint	
C6A6			LD A,(HL)	
C6A6 7E			OR A	
C6A7 B7			RET Z	
C6A8 C8			CP ''	
C6A9 FE20			JR NZ,MSG2	
C6AB 2002			LD A,'0'+10	
C6AD 3E3A				
C6AF		MSG2:	;MSGprn 2	
C6AF D630			SUB 0	
C6B1 FE0B			CP 11	
C6B3 3802			JR C,MSG1	与えられる文字列を返す
C6B5 D606			SUB 6	
C6B7		MSG1:	;MSGprn 1	
C6B7 C5			PUSH BC	
C6B8 E5			PUSH HL	
C6B9 CDB4BF			CALL DISPLE	
C6BC E1			POP HL	
C6BD C1			POP BC	
C6BE 0C			INC C	
C6BF 0C			INC C	
C6C0 23			INC HL	
C6C1 18E3			JR MSGPRN	
			;CPRoad: ;Count Paintable ROAD	
C6C3			LD HL,PAINWK	敵の座標をワークエリアに格納
C6C3 219FC3			LD (HL),0	ワークエリアの初期化
C6C6 3600			LD DE,IMAZE	敵の座標をワークエリアに格納
C6C8 119CC0			LD B,192	敵の座標をワークエリアに格納
C6CB 06C0				
C6CD		CPRLP:	;CPRoad Loop	
C6CD 1A			LD A,(DE)	
C6CE 13			INC DE	
C6CF B7			OR A	
C6D0 2001			JR NZ,SKIPC	敵の座標をワークエリアに格納
C6D2 34			INC (HL)	
C6D3		SKIPC:	;SkIP Count	
C6D3 10F8			DJNZ CPRLP	
C6D5 11A0C3			LD DE,PAINWK+1	
C6D8 010600			LD BC,6	敵の座標をワークエリアに格納
C6DB EDB0			LDIR	敵の座標をワークエリアに格納
C6DD C9			RET	
			;EMMVAL: ;EnEMy MoVe ALl	
C6DE			LD IX,EMWORK	
C6DE DD2121C5			LD B,3	
C6E2 0603				
C6E4		EMALP:	;EmMVAL Loop	
C6E4 C5			PUSH BC	
C6E5 CD30C5			CALL EMMOVE	敵の座標をワークエリアに格納
C6E8 C1			POP BC	
C6E9 110500			LD DE,5	
C6EC DD19			ADD IX,DE	
C6EE 10F4			DJNZ EMALP	
C6F0 C9			RET	
			;MYCHK: ;MY Check	
C6F1			LD HL,EMWORK+3	敵の座標をワークエリアに格納
C6F1 2124C5			LD DE,5	敵の座標をワークエリアに格納
32570 C6F4 110500				


```

C6F7 0603      LD      B,3
C6F9           MCLOOP: ;MyChk : OOP
C6F9 3AA9C3     LD      A,(MYWORK+2)
C6FC 96         SUB     (HL)
C6FD C602       ADD     A,2
C6FF FE05       CP      5
C701 300B       JR      NC,NCRASH
C703 3AAAC3     LD      A,(MYWORK+3)
C706 23         INC     HL
C707 96         SUB     (HL)
C708 2B         DEC     HL
C709 C602       ADD     A,2
C70B FE05       CP      5
C70D 08         RET     C
C70E           NCRASH: ;N : CRASH
C70E 19         ADD     HL,DE
C70F 10E8       DJNZ    MCLOOP
C711 C9         RET

```

```

;
;***** List 5-4-T *****
;

```

```

D000           TEST: ;TEST
D000 F3         DI
D001 AF         XOR     A
D002 D351       OUT     (51H),A
D004 3100B6     LD      SP,STACK
D007 ED5F       LD      A,R
D009 321FC5     LD      (RNDWOK),A
D00C CD88BF     CALL    CLS
D00F CD58C2     CALL    MKIMZ
D012 CD8CC2     CALL    MKAMZ
D015 CDDDC2     CALL    RMEDGE
D018 CDF1C2     CALL    DISPMZ
D01B CDC3C6     CALL    CPROAD
D01E 3A9FC3     LD      A,(PAINWK)
D021 3205C4     LD      (BONSCH+1),A
D024 3E10       LD      A,16
D026 32B4C5     LD      (CFWORK),A
D029 AF         XOR     A
D02A 219CC3     LD      HL,SCORE2
D02D 77         LD      (HL),A
D02E 23         INC     HL
D02F 77         LD      (HL),A
D030 23         INC     HL
D031 77         LD      (HL),A
D032 110000     LD      DE,0
D035 CD6AC3     CALL    DISPSC
D038 01440F     LD      BC,MANPOS
D03B 3E01       LD      A,1
D03D CD10BF     CALL    DISP
D040 3E03       LD      A,3
D042 32ABC3     LD      (MYWORK+4),A

```

```

;
D045           TEST1: ;TEST 1
D045 AF         XOR     A
D046 21A7C3     LD      HL,MYWORK
D049 0604       LD      B,4
D04B           T1LP: ;Test 1 Loop
D04B 77         LD      (HL),A
D04C 23         INC     HL
50411 D04D 10FC   DJNZ    T1LP

```

D04F 2137D1	LD HL,EMINIT
D052 1121C5	LD DE,EMWORK
D055 010F00	LD BC,15
D058 EDB0	LDIR
D05A 3AABC3	LD A,(MYWORK+4)
D05D 014A10	LD BC,RESLOC
D060 CDB4BF	CALL DISPLE
; TEST2:	
D063	IN A,(9)
D063 DB09	LD (SPACE),A
D065 32A6C3	RRCA
D068 0F	JR NC,TEND
D069 306D	CALL EMMVAL
D06B CDDEC6	CALL MYMOVE
D06E CDACC3	LD HL,MYWORK
D071 21A7C3	XOR A
D074 AF	OR (HL)
D075 B6	JR Z,NOMSC
D076 280D	INC HL
D078 23	LD A,(HL)
D079 7E	ADD A,A
D07A 87	ADD A,(HL)
D07B 86	LD E,A
D07C 5F	LD D,0
D07D 1600	LD HL,MMDATA
D07F 2128D1	ADD HL,DE
D082 19	JR CMUSIC
D083 1803	
D085	NOMSC:
D085 2134D1	LD HL,MMDATA+12
D088	CMUSIC:
D088 CD00C0	CALL MUSIC
D08B 3AA6C3	LD A,(SPACE)
D08E E640	AND 40H
D090 28D1	JR Z,TEST2
D092 CDF1C6	CALL MYCHK
D095 3808	JR C,MYDEAD
D097 3AA5C3	LD A,(PAINWK+6)
D09A B7	OR A
D09B 20C6	JR NZ,TEST2
D09D 181D	JR WINNER
D09F	MYDEAD:
D09F 2117D1	LD HL,DMDATA
D0A2 CD34C0	CALL SND1
D0A5 21ABC3	LD HL,MYWORK+4
D0A8 35	DEC (HL)
D0A9 2805	JR Z,GOVER
D0AB CDF1C2	CALL DISPMZ
D0AE 1895	JR TEST1
D0B0	GOVER:
D0B0 AF	XOR A
D0B1 014A10	LD BC,RESLOC
D0B4 CDB4BF	CALL DISPLE
D0B7 21DAD0	LD HL,GOMSG
D0BA 1803	JR MSG
D0BC	WINNER:
D0BC 21EDD0	LD HL,WMSG
D0BF	MSG:
D0BF 010E10	LD BC,100EH
D0C2 CDA6C6	CALL MSGPRN

```

1020 D0C5 21FFD0      LD    HL,PSMSG
      D0C8 010920      LD    BC,2009H
      D0CB CDA6C6      CALL MSGPRN
      D0CE          WLOOP: ;Waiting LOOP
      D0CE DB09          IN    A,(9)
      D0D0 CB77          BIT    6,A
      D0D2 CA00D0        JP     Z,TEST
      D0D5 0F          RRCA
      D0D6 38F6          JR     C,WLOOP
      D0D8          TEND: ;Test END
      D0D8 FB          EI
      D0D9 FF          RST    38H

      ;
      D0DA          GOMSG: ;Game Over Message
51160 D0DA 20472041      DB     ' G A M E
      D0DE 204D2045
      D0E2 2020
      D0E4 4F205620      DB     'O V E R ',0
      D0E8 45205220
      D0EC 00
51180 D0ED          WMSG: ;Winner Message
      D0ED 20434F4E      DB     ' CONGRAT
51190 D0F1 47524154
      D0F5 554C4154      DB     ' ULATIONS ',0
51200 D0F9 494F4E53
      D0FD 2000
51210 D0FF          PSMSG: ;Press Space Message
51220 D0FF 20505245      DB     ' PRESS SPACE
      D103 53532053
      D107 50414345
      D10B 20
51230 D10C 544F2052      DB     ' TO REPLAY ',0
      D110 45504C41
      D114 592000
51240 D117          DMDATA: ;Dead Music DATA      --- 敵突音データ
      D117 14280A00      DB     20,40,10,0
      D11B 14500A00      DB     20,80,10,0
      D11F 14780A00      DB     20,120,10,0
51280 D123 28A0FF00      DB     40,160,255,0,0
      D127 00
51290 D128          MMDATA: ;Move Music DATA      --- 移動音データ
      D128 228700      DB     22H,87H,0
      D128 247800      DB     24H,78H,0
      D12E 286B00      DB     28H,68H,0
      D131 2A6500      DB     2AH,65H,0
      D134 140000      DB     14H,0,0
      D137          EMINIT: ;Enemy Initial data      --- 敵の初期データ
51360 D137 0001003C      DB     0,DU,0,60,0
      D13B 00
51370 D13C 01100000      DB     1,DL,0,0,44
      D140 2C
51380 D141 0204003C      DB     2,DD,0,60,44
      D145 2C
51390          ;
      0F44          MANPOS:EQU 0F44H ;MAN's POSition      一残敵の横の主人公表示座標
51410 104A          RESLOC:EQU 104AH ;RESt number LOCation 一残敵表示座標

```


コラム

BASIC のプログラムは、List を見ながら打ち込み後は RUN で実行するだけで良い。プログラムのミスで、無限ループに陥っても「STOP」を押し、もう一度プログラムをデバッグしてから RUN すれば、良いのである。

しかしマシン語は、そう簡単にはいかない。まず、アセンブラのダンプリストをモニタから入力、チェック・サム・プログラム等により正しく打ち込まれたか確認。もしもまちがっていたら、すぐに暴走してしまう。次にアセンブラのリストを入力し、アセンブルしなくては、ならない。さらに、本書では、各サブルーチンごとに学習していくため、リストのマージや分割アセンブルを駆使するので、オペレーションがめんどろである。そこで、

長大プログラム入力拒否症

長大プログラム入力疲労

アセンブル・アレルギー

膨大データ・チアノーゼ

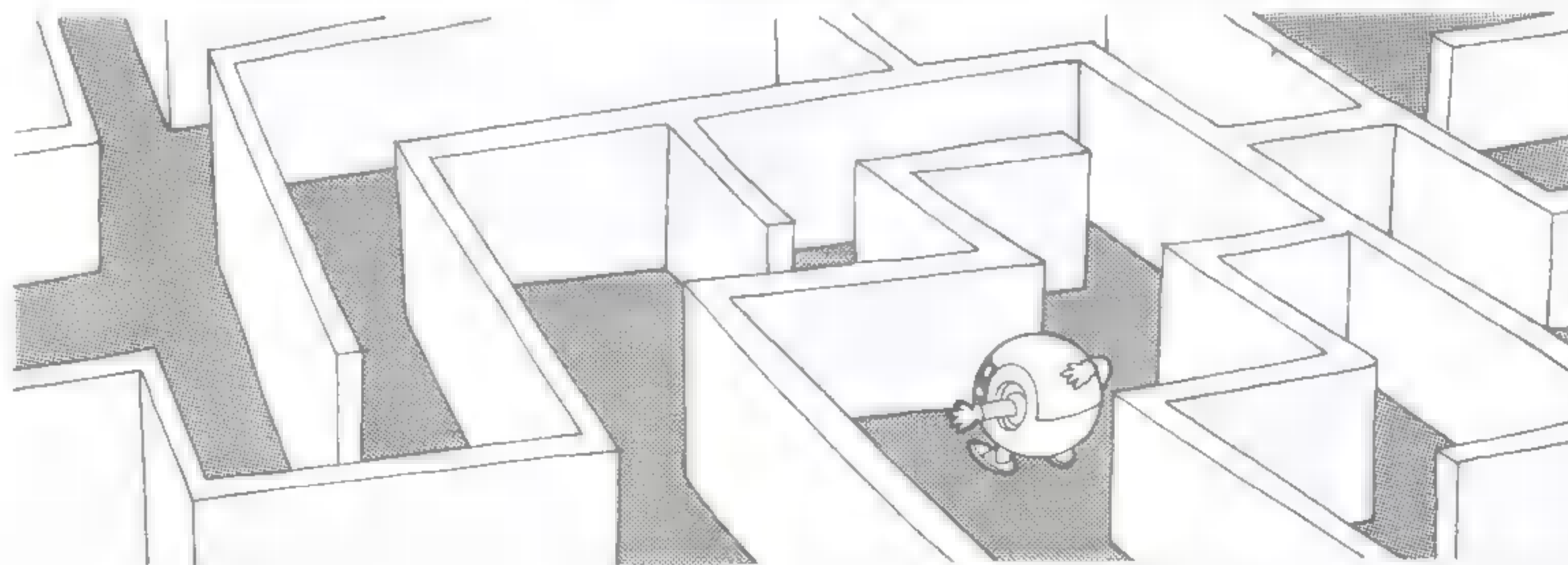
などの併発が予想される。読者の心身を気

遣う我々 Staff は、本書のプログラム、データをすべて入力した上、心ばかりのおまけまでつけたディスクセットを発売する。

アスキー・ディスク・アルバム10として PC-8801mkIISR マシン語ゲーム・プログラミングとタイトルも本書と同じである。レコード・ジャケットを少し小さくした薄型のシースに入って 5800 円。

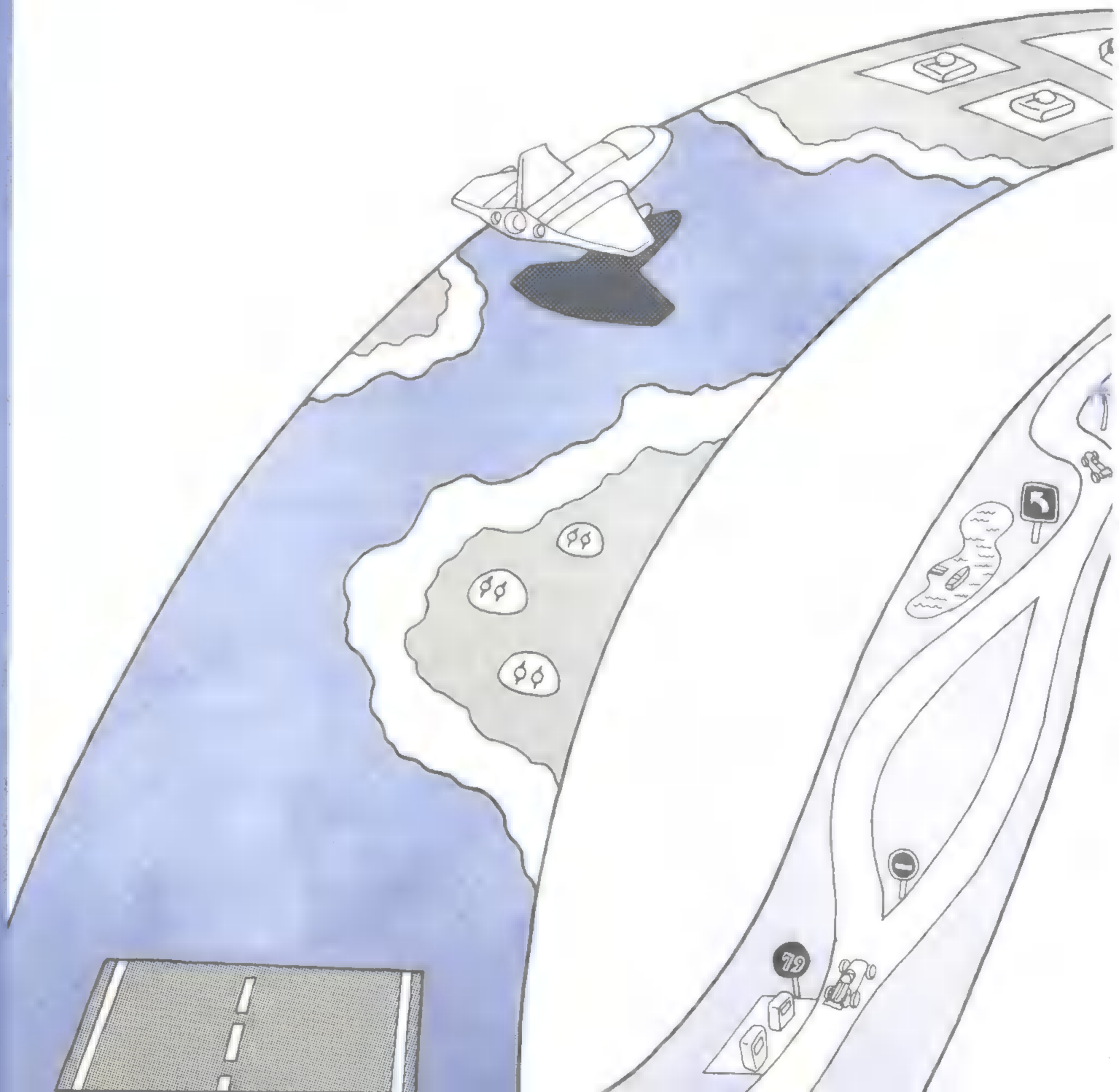
この中に、アセンブラからスクロール・ゲーム「スカイ・ブルーザー」とそのコンストラクション・キット、迷路型ゲーム「ペンキ・ボーイ」、サンプル・シューティング・ゲーム、マップ・エディタまで、約 50 本以上のプログラムが入っている。さらに、スペースのつごう上、本に載せられなかったプログラムも収録。もちろんすべて、自分で入力するに越したことはないが、時間的余裕のない人、めんどろな人には最適であろう…?

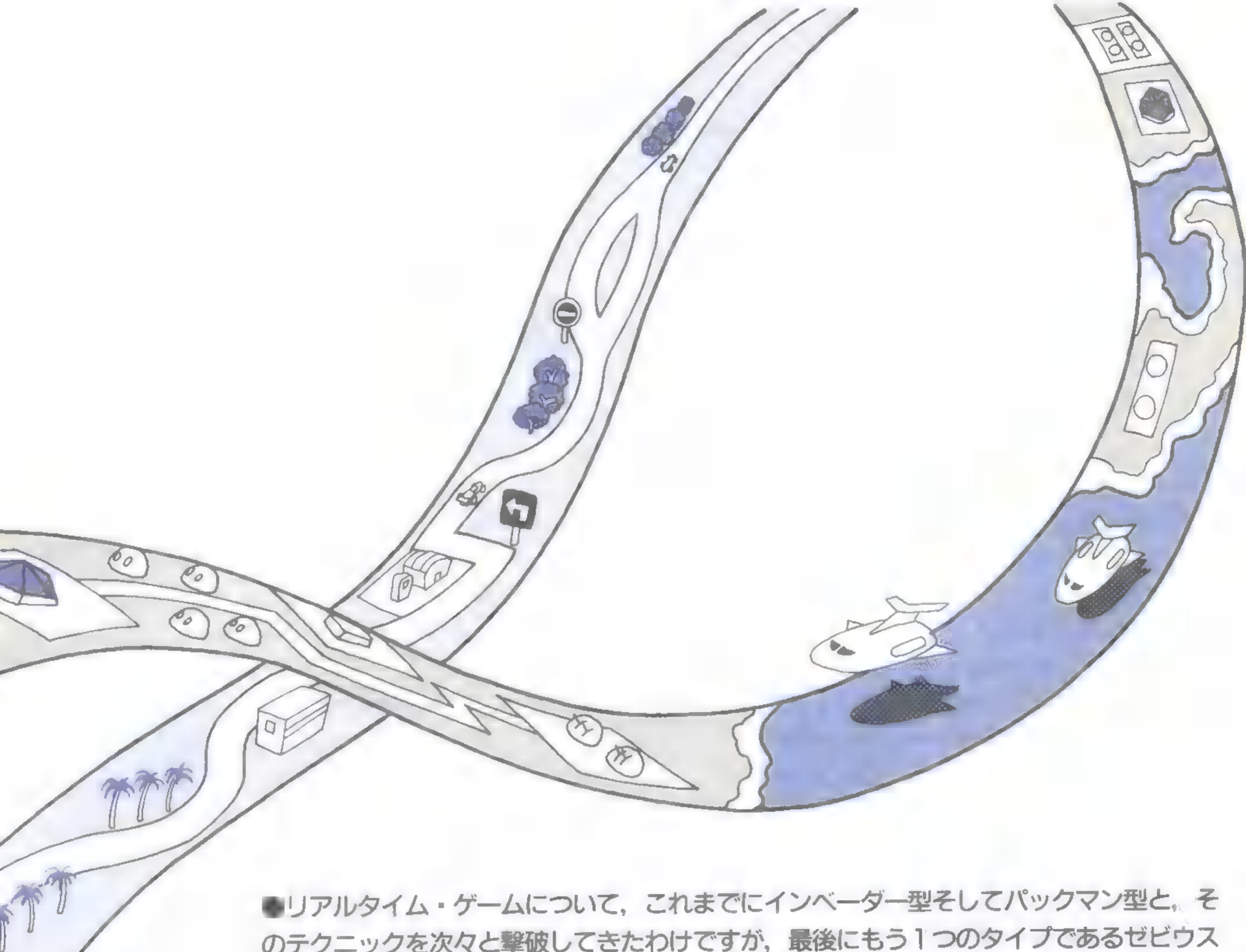
なお、本書や市販のゲーム・ソフトに対する御意見は、アンケート・ハガキに書いて送ってください。編集スタッフの感情を高ぶらせた方には、何か送ってあげよう。



●スクロール・ゲーム

1. 重ね合わせ…もはや一般教養です
2. 割り込み…ウェイトと重ね合わせ
3. QRL…パターン・コントロール言語
4. スカイ・ブルーザー…Playing Game





●リアルタイム・ゲームについて、これまでにインベーダー型そしてパックマン型と、そのテクニックを次々と撃破してきたわけですが、最後にもう1つのタイプであるゼビウス型ゲームがあなたを待っています。何と云っても、スクロール・ゲーム大流行の本家であり、神話までできたと言われるほどの人気ゲームですから、本書でもその存在を無視するわけにはいきません。そのわりに、PC 8801用のスクロール・ゲームは、数多く出ていたとは言えません。本来ならば、「猫も杓子もスクロール…」とばかりに、アチコチのソフト・メーカーから商品化されて良さそうなものですが、一体どうしたのでしょうか。

●PC 8801のようなスクロールに便利な機能がついていないVMEでは、プログラムによってスクロールをさせるしか、その方法がありません。一般的なアルゴリズムのプログラムでスクロールをさせると速度の問題から、無理があると判断せざるを得ないのです。そこで、本章で作るスクロール・ゲームはPC 8801mkⅡSRを使うことを前提としました。エッ！これをPC 8801で実行したらどうなるかって？ やってみる価値は十分あります…少し遅いけどネ！

■さらに、このスクロール・ゲームは、マップやキャラクタ、■の動きなどをあなた自信が作れるように、コンストラクション・セットとして構成してあります。そのため、ディスクがないとマップが作れないので、テープを利用している人…ゴメンナサイ。本来ならソフト・メーカーの極秘事項をプロテクトもかけずソース全公開してしまうという、ゲーム・ソフト史上初の画期的かつ大胆な試みとなりました。

1. 重ね合わせ…もはや一般教養です

これから、画面スクロールというテーマで進むハズなのに、ここに出てきたタイトルは何と『重ね合わせ』です。これは、1つには純粹なスクロールには移動パターンとの重ね合わせ処理が必要であるということからなのですが、もう1つの理由は本格的な重ね合わせテクニックもついでにマスターしてしまおうという、都合のいい理由からです。PC-8801シリーズにもPCG (Programmable Character Generator) やスプライトの機能があれば、このような重ね合わせのことなど気にせずに済むのですが、ないことを嘆いても仕方ありません。これを機会に、重ね合わせのすべてをモノにしていましましょう。

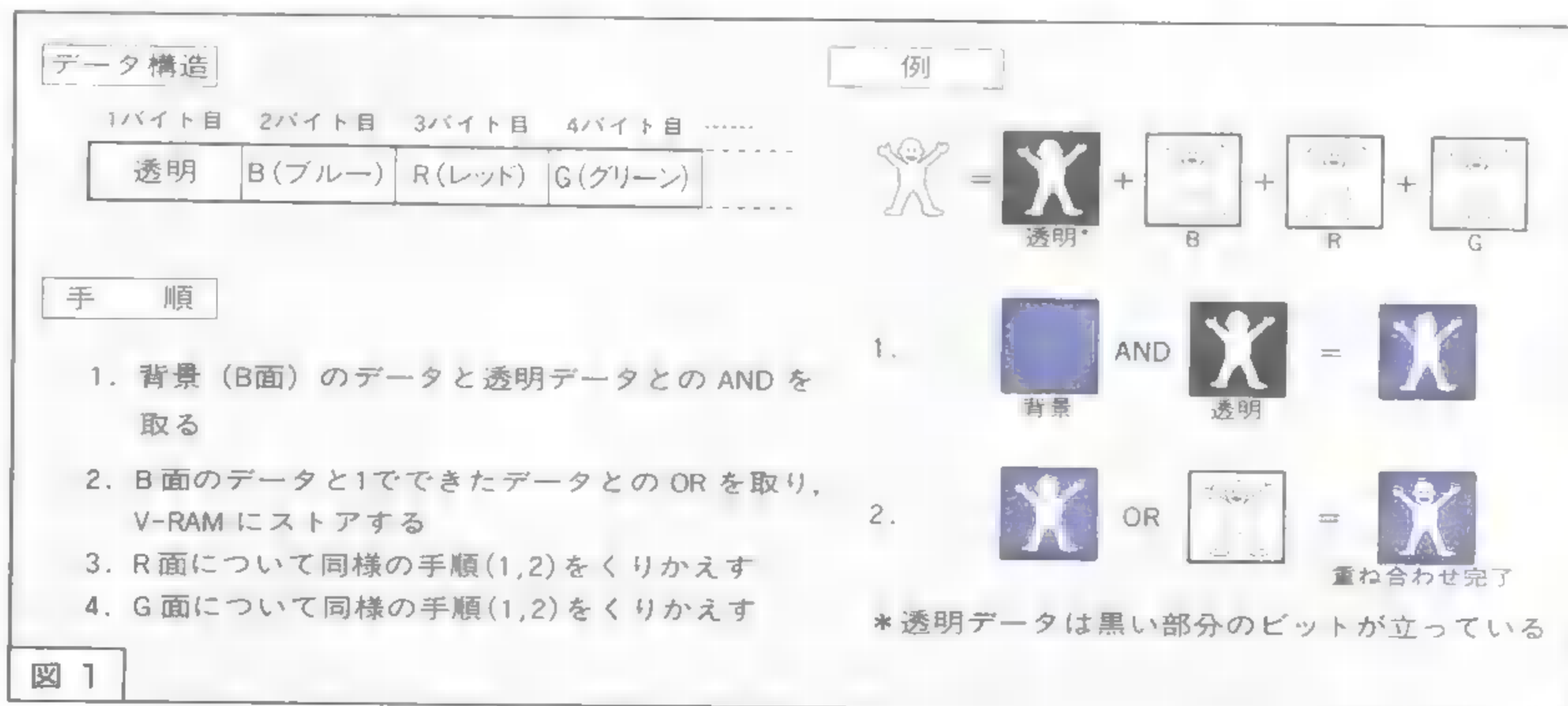
ここに紹介するのは、『完璧な重ね合わせ』です。ただし、本物だけあって少しばかりめんどくさくすし、パターン・データもこれまでとは違い『完璧な重ね合わせ』専用

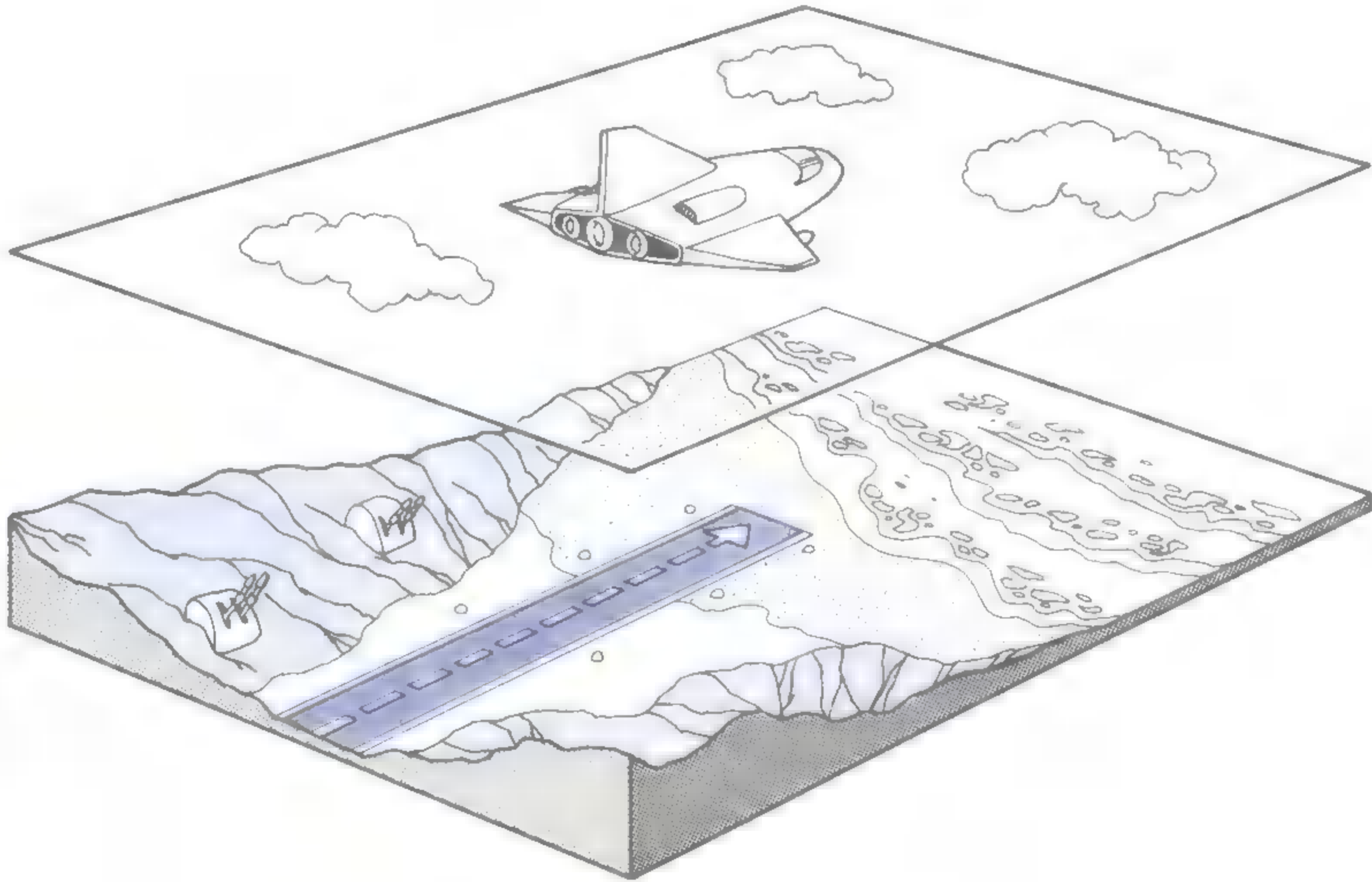
のものがようになります。

その上、パターンを移動させるには、背景となっているB・R・G面のデータを画面とは別に持つ必要があります。工夫すれば、これはデータそのものではなく、背景のパターン番号を示す画面データでもできないことはありませんが、プログラ的にはより複雑になります。

そのために、実際に『完璧な重ね合わせ』を用いたゲームも、あまり多くはありません。この場合、画面処理そのものよりも、どのようにして背景画面のデータを圧縮するかが重要なテーマであるからです。これらのことを頭に入れた上で、『完璧な重ね合わせ』のデータ構成、およびその手順を見てみることにいたしましょう(図1参照)。

まず、このような特殊なデータを作る方法ですが、これはAppendixのパターン・エディタを使えば簡単に作成することができ





ます。つまり、透明(背景)を出したい部分を実際には存在しないパレット番号の8(パターン・エディタのパレット番号)で描いておき、パターン作成終了時に3のデータを選択すればいいのです。これで、希望通りのデータができます。次は、重ね合わせの手順です。図1の例で確認すると、最初に背景と透明部分とのANDを取り、できたデータとその面のパターン・データとのORを取るという作業を、B・R・G各面について行なえば、背景とパターンが完全に重なるというわけです。

このように、背景とパターンをただ重ね合わせるだけであれば、プログラムにするのもそれほどむずかしいことではないし、

ANDを取るのも画面(グラフィック・V-RAM)上のデータでいいのですが、大変なのは重ね合わせたパターンを移動させる時です。この場合、透明データとANDを取るのには、実際に画面上にあるデータではなく、別のメモリに確保してあるものでなければなりません。一見すると、画面上のデータでも良さそうな気がしますが、消去(方向別の不要部分に背景を描く)後に画面との重ね合わせをすると、実際には古いパターンの残骸の一部とも重ね合わせ処理を行なうことになり、結果的には背景が乱れてしまいます。これを避けるには、方向にかぎらずパターン全部を消去(背景を描く)すればいいのですが、そうすると速度的な問題か

らパターンのチラツキがヒドクなるため、あまりお勧めできません。

背景データは、当然のことながら表示の時だけでなく、移動のための消去、すなわち背景を描く時にも使われているわけです。移動パターンが1つしかなければ、画面上のデータをいったんメモリに退避しておくという手も考えられますが、ゲームではいくつものパターンが移動しているので、敵同士が重なる場合もでてきます。そのため、この方法では正確な背景のデータを保存することはできません。また、この方法で部分消去を行なうには、プログラムも異常に複雑になり実際には頭がこんがらかるような処理になってしまいます。そのため、移動パターンは1つ、方向は固定、速度の追求はしない、という条件つきでないと、現実には不可能といえます。

以上のような理由から、どうしても画面上のデータをどこかに確保しておかなければならないのですか、今度はメモリのフリー・エリアの問題があります。つまり、もし背景をフル・カラーで画面一杯に描くとすると、それだけで48Kバイトものメモリを占めることになり、これまた非現実的ということになってしまいます。

ある有名なゲームでは、この問題を解決するために、背景に使う色を4色に限定し、しかも1ドットおきに描くということをしています。こうすると、使用するプレーン(面)は2面で済む上に、1バイトの中に両プレーンのデータを交互に入れておくことができるわけです。したがって、背景のためのデータは単純計算で、1/3の16Kで間に合うことになります。しかし、ここま

でしても描ける背景はせいぜい1種類で、後はパレットによる色変化しか楽しみはありません。これは、画面を完全な1枚の絵としているからで、それほど頑張らずに同一パターンをいくつも使用して背景にすれば、何面もの画面を持つことができます。この辺の判断は、作者の好みとそのゲームでの特徴というこになりますので、どちらが優れているということではありません。いずれにしても背景データの圧縮という問題が、「完璧な重ね合わせ」テクニックのキー・ポイントであることは間違いのないところです。

さて、ここであまり凝ったことをしても、この章の本質から離れてしまいますから、サンプル・プログラムの内容としては2章の5節程度のもの、とします。そして、背景は0000H番地からBFFFH番地にあるROM, RAM合わせて48Kバイトの内容を、B, R, G, B, R, G, …と続く背景のデータと見なすことにしました。現実には、たとえフルRAM・モードであっても、このように画面全体のデータを持つことなど不可能ですが、ここでは実践的な重ね合わせとして、その処理方法の基礎を理解するつもりで、List 6-1のプログラムを見てください。

このList 6-1は、基本的にはList 2-5に背景(というより、キタナイ縞模様)をつけているだけなのですが、重ね合わせ処理の部分ではレジスタ数が足りず、裏レジスタも使用しています。そのため、プログラムの内容が少しわかりにくくなっているかもしれませんが、各レジスタの役目を把握すれば、B, R, G面同じことをしているので、それほど複雑ではないと思います。

プログラムのテストに際しては、重ね合わせ専用のパターン・データを用いないと、その意味がありませんから、カラーページの4のパターンを参考にし専用データを作成してから、実行してください。

何度も繰り返すようですが、この『完璧

な重ね合わせ』の欠点は、背景データにメモリを使い過ぎることです。これを解決するには、何らかの工夫とアイディアが必要なのですが、先ほど少し触れた同一パターンを多用するという方法について、その概略をまとめてみます。

1. パターンの移動単位は2コマとする。
2. 背景パターンのサイズは2×2コマ(16×8ドット)を基準とする。
3. 背景のデータはパターン番号で表わす。…背景番号

これらの条件を背景および移動パターンに付けることにより、これまでのように1ドット単位での背景データは持つ必要はなくなります。もちろん、背景パターン総数分のデータはなければなりませんが、画面全体を持つことに比べれば月とスッポンです。また、パターンの移動コマ数が、背景パターン2つ(上下左右の場合)にしてあるので、プログラムを組む上で消去処理が大幅に楽になります。さらに、パターン移動時に次座標の背景番号を見ることにより、これを座標データとしても利用できる

ので、画面の迷路化が簡単に行なえることにもなるのです。

ただし、このような例はゲームの内容がその条件を満たせるということが前提であり、どんなゲームもこれで良いというわけにはいきません。

それにしても、完璧とは何ともめんどうなことです。もう少し、何とかならないものでしょうか。そこでスクロール専用ともいうべき完璧で簡単な重ね合わせの方法があるので…

List 6-1 重ね合わせ例

```

;***** List 6-1 *****
;
B600    STACK: EQU 0B600H ;STACK pointer
C000    VTOP:  EQU 0C000H ;Vram TOP address
0050    HLEN:  EQU 80     ;Horizontal LENgth
00F0    HLEN3: EQU 240    ;HLEN * 3
0140    HLEN4: EQU 320    ;HLEN * 4
;
;      ORG 0BE00H
;
BE00    KASANE: ;KASANE awase
BE00    CD7FBE   CALL XYADR
BE03    ED5B48BE LD DE,(DISPAD)
BE07    2100B6   LD HL,0B600H
10140   BE0A 011004 LD BC,410H

```

…表示アドレス
 …表示アドレス
 …表示サイズ

10150	BE00 D9	EXX	
	BE0E 2A40BE	LD HL,(DATAAD)	
	BE11 11E400	LD DE,HLEN3-12	
	BE14 D9	EXX	
	BE15	KL1: ;Kasane aware Loop 1	
	BE15 C5	PUSH BC	
	BE16 D5	PUSH DE	
	BE17	KL2: ;Kasane aware Loop 2	
	BE17 D35C	OUT (5CH),A	
	BE19 7E	LD A,(HL)	
	BE1A 4F	LD C,A	
	BE1B D9	EXX	
	BE1C A6	AND (HL)	
	BE1D 23	INC HL	
	BE1E D9	EXX	
	BE1F 23	INC HL	
	BE20 B6	OR (HL)	
	BE21 12	LD (DE),A	
	BE22 D35D	OUT (5DH),A	
	BE24 23	INC HL	
	BE25 79	LD A,C	
	BE26 D9	EXX	
	BE27 A6	AND (HL)	
	BE28 23	INC HL	
	BE29 D9	EXX	
	BE2A B6	OR (HL)	
	BE2B 12	LD (DE),A	
	BE2C D35E	OUT (5EH),A	
	BE2E 23	INC HL	
	BE2F 79	LD A,C	
	BE30 D9	EXX	
	BE31 A6	AND (HL)	
	BE32 23	INC HL	
	BE33 D9	EXX	
	BE34 B6	OR (HL)	
	BE35 12	LD (DE),A	
	BE36 23	INC HL	
	BE37 13	INC DE	
	BE38 100D	DJNZ KL2	
	BE3A EB	EX DE,HL	
	BE3B E1	POP HL	
	BE3C 015000	LD BC,HLEN	
	BE3F 09	ADD HL,BC	
	BE40 EB	EX DE,HL	
	BE41 D9	EXX	
	BE42 19	ADD HL,DE	
	BE43 D9	EXX	
	BE44 C1	POP BC	
	BE45 0D	DEC C	
	BE46 20CD	JR NZ,KL1	
	BE48 D35F	OUT (5FH),A	
	BE4A C9	RET	
	BE4B	DISPAD: ;DISPlay Address	
	BE4B	DS 2	
	BE4D	DATAAD: ;DATA Address	——背景データアドレス
	BE4D	DS 2	
	BE4F	DBACK: ;Display BACKground	——背景の表示
	BE4F E5	PUSH HL	
10760	BE50 CD7FBE	CALL XYADR	

```

BE53 EB      EX    DE,HL
BE54 2A4BBE   LD    HL,(DISPAD)
BE57 C1      POP   BC
DBLP1:      ;DBack Loop 1
BE58 C5      PUSH  BC
BE59 E5      PUSH  HL
BE5A D5      PUSH  DE
DBLP2:      ;DBack Loop 2
BE5B D35C    OUT   (5CH),A
BE5D 1A      LD    A,(DE)
BE5E 77      LD    (HL),A
BE5F D35D    OUT   (5DH),A
BE61 13      INC   DE
BE62 1A      LD    A,(DE)
BE63 77      LD    (HL),A
BE64 D35E    OUT   (5EH),A
BE66 13      INC   DE
BE67 1A      LD    A,(DE)
BE68 77      LD    (HL),A
BE69 13      INC   DE
BE6A 23      INC   HL
BE6B 10EE    DJNZ  DBLP2
BE6D E1      POP   HL
BE6E 01F000   LD    BC,HLEN3
BE71 09      ADD   HL,BC
BE72 EB      EX    DE,HL
BE73 E1      POP   HL
BE74 015000   LD    BC,HLEN
BE77 09      ADD   HL,BC
BE78 C1      POP   BC
BE79 0D      DEC   C
BE7A 20DC    JR    NZ,DBLP1
BE7C D35F    OUT   (5FH),A
BE7E C9      RET

```

```

;
;XYADR:      ;XY to ADdRes
BE7F 68      LD    L,B
BE80 2600    LD    H,0
BE82 29      ADD   HL,HL
BE83 29      ADD   HL,HL
BE84 29      ADD   HL,HL
BE85 29      ADD   HL,HL
BE86 29      ADD   HL,HL
BE87 29      ADD   HL,HL
BE88 09      ADD   HL,BC
BE89 E5      PUSH  HL
BE8A E5      PUSH  HL
BE8B 0100C0   LD    BC,VTOP
BE8E 09      ADD   HL,BC
BE8F 224BBE   LD    (DISPAD),HL
BE92 E1      POP   HL
BE93 C1      POP   BC
BE94 29      ADD   HL,HL
BE95 09      ADD   HL,BC
BE96 224DBE   LD    (DATAAD),HL
BE99 C9      RET

```

```

;
;MVCLS:      ;MoVe CLS
BE9A C5      PUSH  BC
BE9B 3D      DEC   A
BE9C 2825    JR    Z,D1CLS
BE9E 3D      DEC   A

```

DE = 2A4B (DISPAD) に
LD HL, (DISPAD)
POP BC

画面について、背景データを
メモリ (グラフィック V-RAM) にストア

描画データをメモリにだけ繰り返す

描画データをメモリにだけ繰り返す

描画データの表示アドレス

描画データをメモリにだけ繰り返す

描画データの V-RAM アドレスを
求める
描画データの V-RAM アドレス
を求める
描画データの V-RAM アドレス
を求める
描画データの V-RAM アドレスは 0000H 番地で
メモリに格納されている
描画データの V-RAM アドレスは (グラフィッ
ク V-RAM) に格納されている

BE9F 2828	JR Z,D2CLS
BEA1 3D	DEC A
BEA2 283D	JR Z,D3CLS
BEA4 3D	DEC A
BEA5 2846	JR Z,D4CLS
BEA7 3D	DEC A
BEA8 285B	JR Z,D5CLS
BEAA 3D	DEC A
BEAB 2864	JR Z,D6CLS
BEAD 3D	DEC A
BEAE CA27BF	JP Z,D7CLS
;D8CLS: ;Direction 8 CLS	
BEB1	LD HL,404H
BEB1 210404	CALL DBACK
BEB4 CD4FBE	POP BC
BEB7 C1	INC B
BEB8 04	PUSH BC
BEB9 C5	LD HL,10CH
BEBA 210C01	CALL DBACK
BEBD CD4FBE	POP BC
BEC0 C1	INC C
BEC1 0C	RET
BEC2 C9	
;D1CLS: ;Direction 1 CLS	
BEC3	LD HL,110H
BEC3 211001	CALL DBACK
BEC6 CD4FBE	POP BC
BEC9 C1	INC C
BECA 0C	RET
BECB C9	
;D2CLS: ;Direction 2 CLS	
BECC	LD HL,10CH
BECC 210C01	CALL DBACK
BECF CD4FBE	POP BC
BED2 C1	PUSH BC
BED3 C5	INC B
BED4 04	INC B
BED5 04	INC B
BED6 04	LD HL,404H
BED7 210404	CALL DBACK
BEDA CD4FBE	POP BC
BEDD C1	DEC B
BEDE 05	INC C
BEDF 0C	RET
BEE0 C9	
;D3CLS: ;Direction 3 CLS	
BEE1	INC B
BEE1 04	INC B
BEE2 04	INC B
BEE3 04	LD HL,404H
BEE4 210404	CALL DBACK
BEE7 CD4FBE	POP BC
BEEA C1	DEC B
BEEB 05	RET
BEEC C9	
;D4CLS: ;Direction 4 CLS	
BEED	INC B
BEED 04	INC B
BEEE 04	INC B
BEEF 04	

```

BEF0 210404      LD    HL,404H
BEF3 CD4FBE      CALL  DBACK
BEF6 C1          POP   BC
BEF7 C5          PUSH  BC
BEF8 0C          INC   C
BEF9 0C          INC   C
BEFA 0C          INC   C
BEFB 210C01      LD    HL,10CH
BEFE CD4FBE      CALL  DBACK
BF01 C1          POP   BC
BF02 05          DEC   B
BF03 0D          DEC   C
BF04 C9          RET

;
; D5CLS:
BF05             INC   C
BF05 0C          INC   C
BF06 0C          INC   C
BF07 0C          LD    HL,110H
BF08 211001      CALL  DBACK
BF0B CD4FBE      POP   BC
BF0E C1          DEC   C
BF0F 0D          RET
BF10 C9

;
; D6CLS:
BF11             LD    HL,404H
BF11 210404      CALL  DBACK
BF14 CD4FBE      POP   BC
BF17 C1          PUSH  BC
BF18 C5          INC   C
BF19 0C          INC   C
BF1A 0C          INC   C
BF1B 0C          INC   C
BF1C 04          INC   B
BF1D 210C01      LD    HL,10CH
BF20 CD4FBE      CALL  DBACK
BF23 C1          POP   BC
BF24 04          INC   B
BF25 0D          DEC   C
BF26 C9          RET

;
; D7CLS:
BF27             LD    HL,404H
BF27 210404      CALL  DBACK
BF2A CD4FBE      POP   BC
BF2D C1          INC   B
BF2E 04          RET
BF2F C9

;
; TEST:
D000             DI
D000 F3          LD    SP,STACK
D001 3100B6      XOR   A
D004 AF          OUT   (51H),A
D005 0351        LD    HL,50C8H
D007 21C850      LD    BC,0
D00A 010000      CALL  DBACK
D00D CD4FBE      LD    BC,1914H
D010 011419

```

D013		TINIT: ;Test INITIALize	
D013 213ED0		LD HL,DATA	移動方向を示すデータ
D016 223CD0		LD (DATAWK),HL	
D019		;TLOOP: ;Test LOOP	
D019 213CD0		LD HL,DATAWK	
D01C 34		INC (HL)	
D01D 2A3CD0		LD HL,(DATAWK)	
D020 7E		LD A,(HL)	
D021 B7		OR A	
D022 28EF		JR Z,TINIT	
D024 CD9ABE		CALL MVCLS	不要部分の
D027 C5		PUSH BC	
D028 CD00BE		CALL KASANE	重ねを
D02B 0E30		LD C,30H	
D02D		COUNTC: ;COUNTER C	
D02D 06FF		LD B,0FFH	
D02F		COUNTB: ;COUNTER B	
D02F 10FE		DJNZ COUNTB	
D031 0D		DEC C	
D032 20F9		JR NZ,COUNTC	
D034 C1		POP BC	
D035 DB09		IN A,(9)	
D037 1F		RRA	
D038 38DF		JR C,TLOOP	
D03A FB		EI	押されていない
D03B FF		RST 38H	
D03C		;DATAWK: ;DATA Work area	
D03C		DS 2	
0001		RR: EQU 1	
0002		UR: EQU 2	
0003		UU: EQU 3	
0004		UL: EQU 4	
0005		LL: EQU 5	方向番号の
0006		DL: EQU 6	
0007		DD: EQU 7	
0008		DR: EQU 8	
D03E		;DATA: ;direction DATA	
D03E 07070708		DB DD,DD,DD,DR	移動方向を示すデータ
D042 07070807		DB DD,DD,DR,DD	
D046 08080801		DB DR,DR,DR,RR	
D04A 08080108		DB DR,DR,RR,DR	
D04E 01010102		DB RR,RR,RR,UR	
D052 01010201		DB RR,RR,UR,RR	
D056 02020203		DB UR,UR,UR,UU	
D05A 02020302		DB UR,UR,UU,UR	
D05E 03030304		DB UU,UU,UU,UL	
D062 03030403		DB UU,UU,UL,UU	
D066 04040405		DB UL,UL,UL,LL	
D06A 04040504		DB UL,UL,LL,UL	
D06E 05050506		DB LL,LL,LL,DL	
D072 05050605		DB LL,LL,DL,LL	
D076 06060607		DB DL,DL,DL,DD	
D07A 06060706		DB DL,DL,DD,DL	
D07E 0700		DB DD,0	

2. 割り込み…ウェイトと重ね合わせ

スクロールのための章なのに、スクロールに直接関係のない重ね合わせに、かなり時間(ページ)を費やしてしまいました。しかし、これも自然の成り行きでそうってしまったわけですから、無理に手を抜くこともできなかったのです。大体が、本書はもともとこの半分位の内容で仕上げる予定だったのですが、書いている内に『これも、あれも…』ということになり、段々中身がふくらんでいったしまったというのか、偽らざる実情なのです。こうなったのも、考えてみると特に誰の責任というのでもなく、やはりこの本自体がそうなるべき運命を最初から持っていたのかもしれませんが。

しかし、それにしてもこのスクロール・ゲームは長いものになってしまいました。とても MF-ASM2 1回だけで、アセンブルすることはできません。そのために、このプログラムは全体を3分割して、それぞれ別々にアセンブルしていきます。アセンブルしてメモリに落ちたものは、いったん

ディスクなりテープにセーブし、最後のプログラムがアセンブルし終えた時点で、最初の2つをロードするようになっていきます。この方法によれば、MF-ASM2 でも相当大きなプログラムを組むことが可能になります。プログラムが増えるたびに別のプログラムのサブルーチンを使うには、そのアドレスを EQU 命令により絶対番地で指定しなければなりません。ですから、1つ目のプログラムにバグがあったり修正をした場合は、2つ目、3つ目のプログラムにも影響をおよぼすことになります。いくらでも長くできるからといって、あまり調子に乗ってプログラムを増やすと、後で大変なことになるというわけです。

さて、これから作るプログラムは、その3分割の第一弾ですが、主な内容としては背景のスクロール、主人公の移動、弾の発射、割り込みによる正確なウェイトとなっています。これを見ると、目新しいのは背景のスクロールとウェイトで、残りは2章で

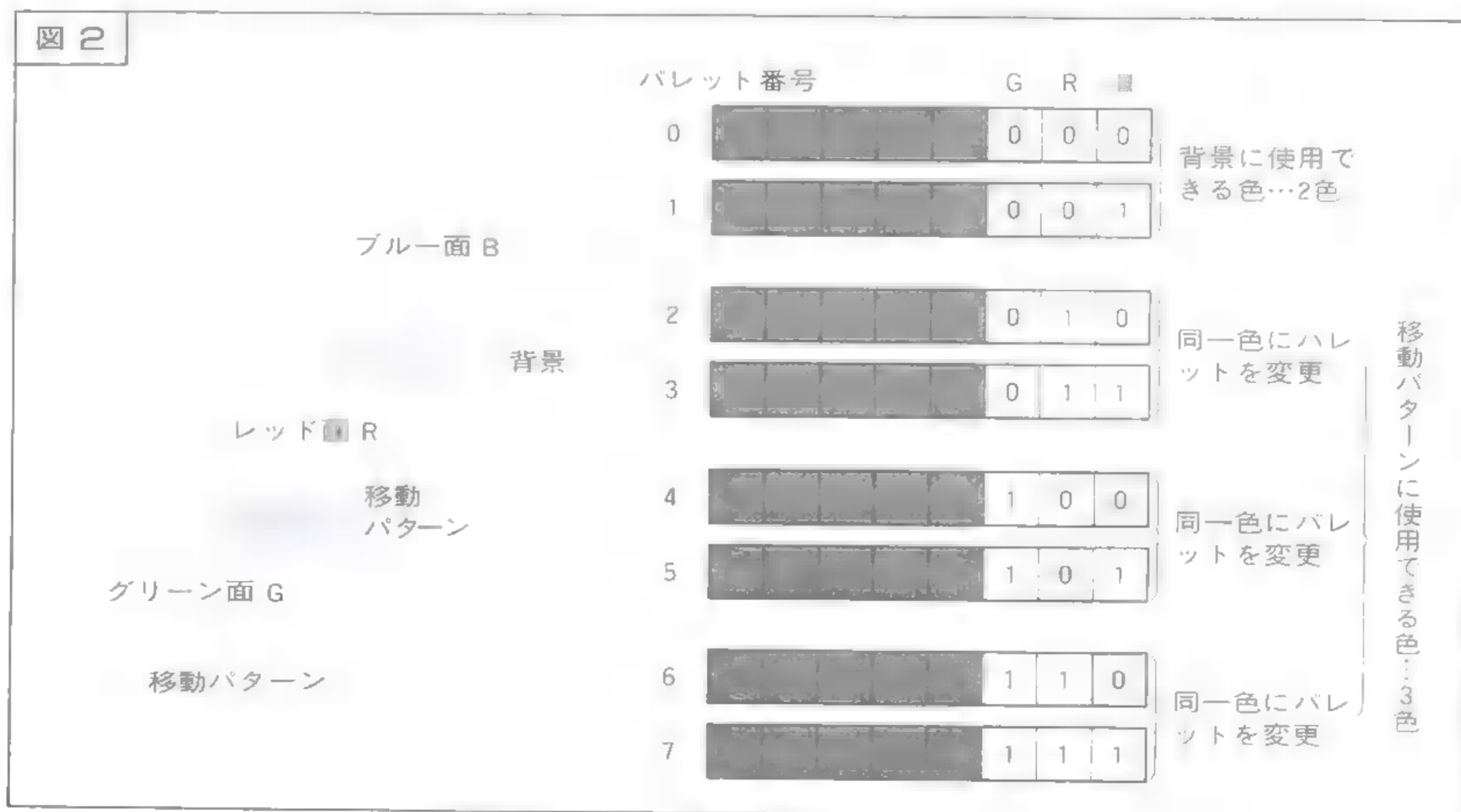


やったことと同じようなものであることがわかります。これまでのものがプログラムで処理していたのに対し、どちらかというハードウェアの特性を利用したものとすることができます。これは、すでに市販のゲームを見て、その色の特徴から原理を知っている方が多いかもしれませんが、B、R、Gの3つのプレーン(面)を背景用と移動パターン用とに分ける方法なのです。

では、下の図2を見て下さい。ここでは、ブルー面を背景用、レッド・グリーン面を移動パターン用として分けています。その結果、背景は2色でしか表現することができません。その2色を何色にするかはパレットの変更でどうにでもなります。移動パターンについては、背景の色がパターンに影響を与えないように、パレット番号で2と3、4と5、6と7をそれぞれ同じ色にパレットを変更をしなければなりません。

これで、ブルー面のビットが1でも0でも、パターンの色は変化しなくなるわけです。そのため、パターンで使える色は3色に減ってしまいますが、レッド面、グリーン面のデータが共に0の場合は、背景の色がそのまま出ることになるので、プログラムの重ね合わせ処理をせずに「完璧な重ね合わせ」が実現できることになるのです。その上、消去の際も従来のように、単純にレッド・グリーン面に0を入れていけば、自動的に背景が出てきます。前の重ね合わせで苦労していたのがウソのような話です。

これだけのメリットがあれば、たとえ使用できる色の数が減っても、価値があるというものです。逆に、この方法以外の重ね合わせ処理ではスクロールと組み合わせることはできないと思って間違いないでしょう。というのは、この重ね合わせでは背景には1面、パターンでも2面しかデータを入



れないで済んでいるのですが、それでも全面スクロールでは速度的にあまり余裕がないというのが実情です。PC-8801mk II SRでは、さすがに波を打つといこうはありませんが、PC-8801では完全に波打ちスクロールになってしまいます。そのため、背景の模様をなるべく均一化して、スクロールしても背景が変化しない部分を多くすることが必要です。こうすれば、スクロールといっても同じ背景パターンのところは描かずに済み、動いて変化する部分だけを描けばいいからです。

スクロールについて、もう1つの重要な点はウェイトをどうするかということです。これまでのゲームであれば、たとえ敵がやられて減ったとしても、敵の総数さえ決まっていれば、それに応じた無駄命令で、ある程度の速度調節が可能でした。今度は、スクロールする地形によっても、描き直す■が違って来るわけなので、より細かくウ

ェイトをかけないと、スクロール速度が変わって見苦しくなってしまいます。そこでこの割込みがかかるとCPUは、それまで実行していた仕事をやめ割込みの処理を先にかたづけます。割込み処理が終わるとまたもとの仕事をやめたところから実行するのです。

List 6-2のプログラムでは…VRTC割込みにより、正確なウェイトをかけてあります。これは、CRTが周期的(1秒間に60回の割合)に画面表示を繰り返しており、画面表示が終了するたびに信号が画面トップに戻るということを利用しています。この垂直帰線期間の回数を、割込みによってカウントし、メイン・ループの中で常に一定の回数になるようにしたのが、ここでのウェイトなのです。

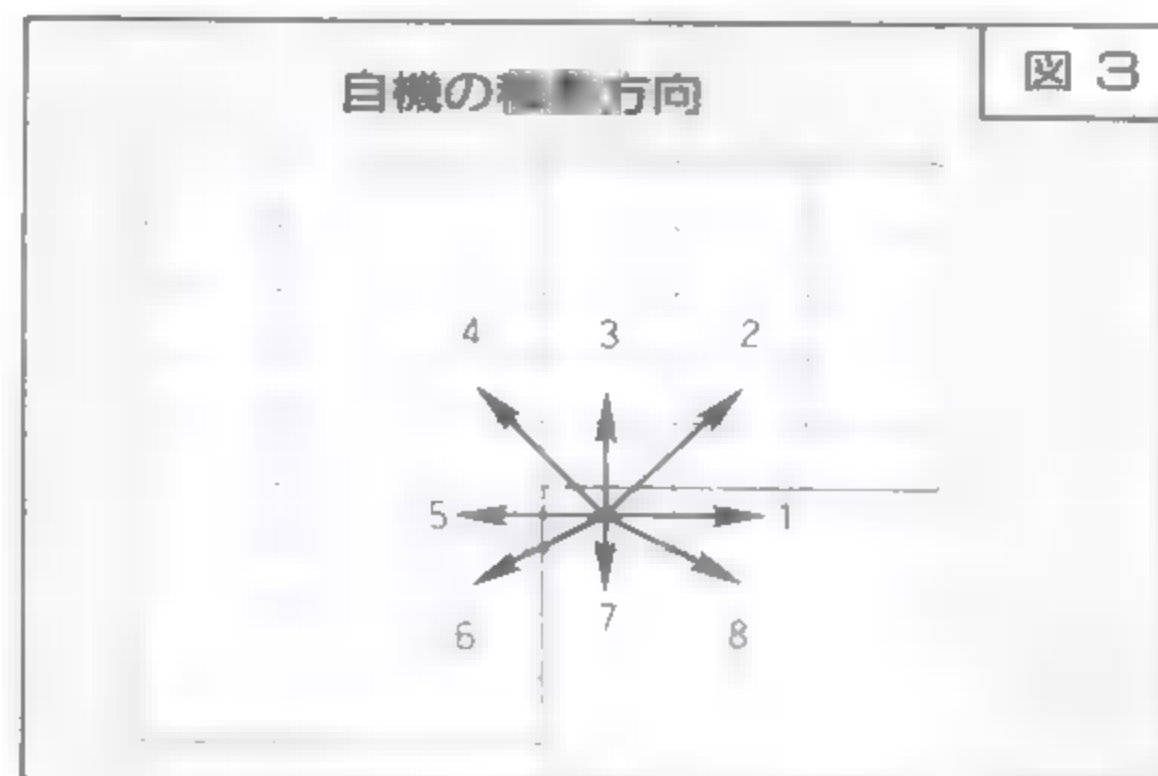
N₈₈-BASICの割込みテーブルは、下のようになっていますから、このままではグラフィック・V-RAMにバンク切り換え中に

表 1

優先順位	アドレス	アドレス・テーブルの内容(ジャンプ先)	チャンネル	用 途	割込みレベル
<div style="display: flex; align-items: center;"> <div style="writing-mode: vertical-rl; text-orientation: upright; margin-right: 5px;">↑</div> <div style="writing-mode: vertical-rl; text-orientation: upright;">上位</div> <div style="writing-mode: vertical-rl; text-orientation: upright; margin-left: 5px;">↓</div> <div style="writing-mode: vertical-rl; text-orientation: upright;">下位</div> </div>	F300	下位アドレス	RXRDY	RS-232C 受信割込み	0
	F301	上位アドレス			
	F302	下位アドレス	VRTC	画面終了割込み	1
	F303	上位アドレス			
	F304	下位アドレス	CLOCK	リアルタイム・クロック(1/600S)	2
	F305	上位アドレス			
	F306	下位アドレス	INT4	ユーザー割込み	3
	F307	上位アドレス			
	F308	下位アドレス	INT3	ユーザー割込み	4
	F309	上位アドレス			
	F30A	下位アドレス	INT2	ユーザー割込み	5
	F30B	上位アドレス			
	F30C	下位アドレス	FDINT1	FDD 用リザーブ	6
	F30D	上位アドレス			
	F30E	下位アドレス	FDINT2	FDD 用リザーブ	7
	F30F	上位アドレス			

割込みが発生すると、ジャンプ先のプログラムがなくなり暴走することになります。そのために、これまでは割込みを禁止していたのですが、ここでは割込みのジャンプ・テーブルを BD00H 番地に設定し直して、それを防いでいます。I(Interrupt)レジスタとは、割込みテーブルの上位アドレスを示すレジスタで、ここに BDH を入れることにより、割込みテーブル・アドレスを BD00H からに変更することができるのです。そして、VRTC 以外の割込みは不要ですから、すべてマスクをかけて禁止します。また、割込みには優先順位があり、優先上位の割込みがあると下位の割込みは入れなくなります。VRTC 割込みは、レベル 1 になっていますから現在の割込みレベルが 2 以下(数字上は 2~7)でないと、割込みがかからなくなってしまいます。割込み処理ルーチンで、毎回毎回割込みレベルを 2 に設定しているのはそのためです。

割込みというのは、実際にいつ、かかるかわかりませんから、割込み処理ルーチンではレジスタを退避しておかないと、元のプログラムに復帰した時に、正常に動作しなくなってしまいます。ここでは、レジスタを使用することによりレジスタの保存をしていますので、本プログラムではレジスタの使用はできません。また、割込みからの復帰はスタック・ポインターを利用しますので、これまでのように SP を単なるレジスタとして使うこともできないわけです。割り込み回数(WTIMES)については、現在 6 に設定していますが、SR でより高速なゲームをしたい場合は、まだ余裕がありますので数を減らしても大丈夫です。



さて、これまでゲーム座標ということ、横 8 ドット・縦 4 ドットを 1 ブロックとしていましたが、このスクロール・ゲームでは、なめらかな画面スクロールを実現するために、背景を縦 2 ドット単位で動かしています。そのために、座標も横 8 ドット・縦 2 ドットを 1 コマとした変則的なものに変えてあります。座標からグラフィック・V-RAM に変換するルーチン(XYADR)や、敵との衝突チェック(MYCHK)では、そのあたりを頭に入れてプログラムを見て下さい。また、背景がスクロールするという事で、上図のように移動方向がこれまでと違い背景を考慮した相対的なものになっています。つまり、座標上で同じ位置にいるということは、背景(地面)から見れば 2 ドットずつ前進しているということになるので、それを移動の計算に入れるということです。これを無視すると、何のためのスクロールかわからなくなってしまいますので、ゲームとして不自然にならない程度に(爆発時などは無視)取り入れてあります。

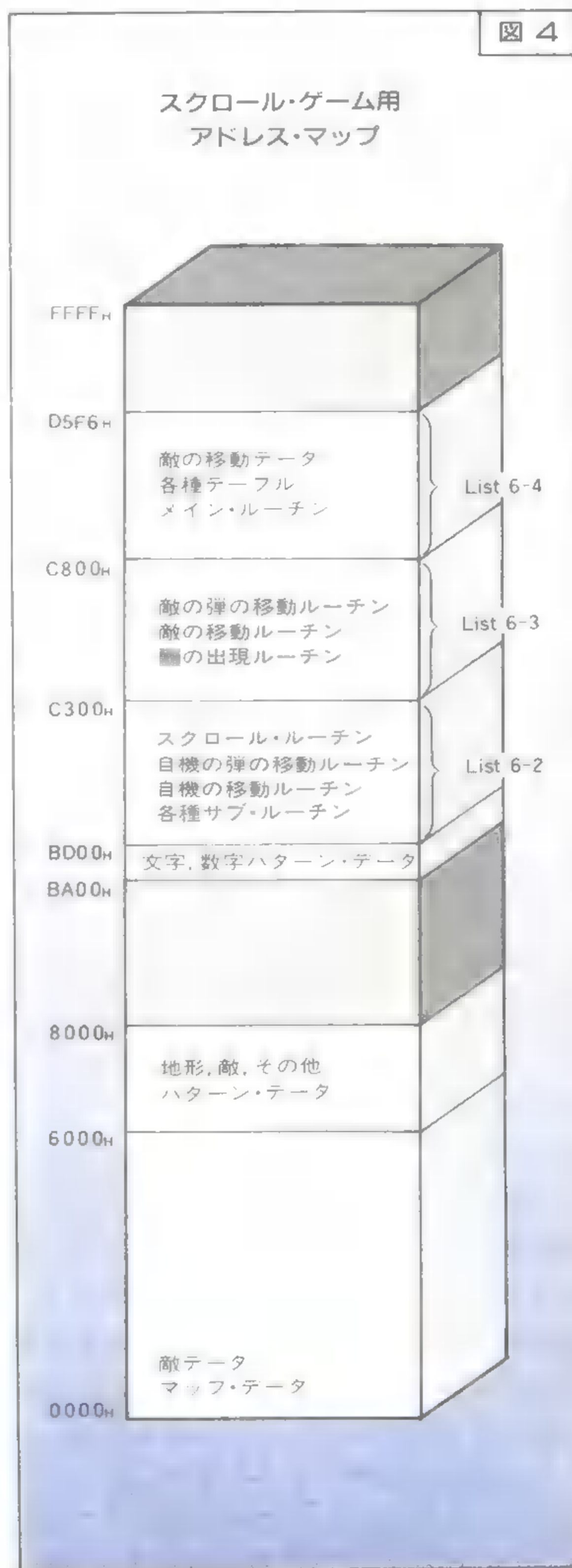
自機に関しての特徴では、SSKCK でスペースバーまたはシフトキーによる弾の連射を可能にさせています。ただし、単に連射を許したのでは、左手はスペースバーを

押しっ放し、あるいはキーの上にガムテープを貼る、などというイカサマを平気で考えるのが人間の常というものです。こうなると、弾の発射はキーによらずに、自動的に出すのと全く同じになってしまいます。そこで、押しっ放しによる連射の場合は、一定の間隔を置くようにして、これを防ぐことにしました。この連射間隔は、キーを押していない時に発生しているデータをカウンターとして利用していますから、プログラ的には簡単な処理で済んでいます。たかが弾の発射に、それほど気を使うこともないと思われる方もいるかもしれませんが、これはゲーム性というより、キーの叩き過ぎによるキーボードの故障を防止するという意味からです。

この List 6-2 には、スクロール・ゲームの全プログラムを3分割したために、仕方なくここに入ってしまった部分(MAKE SC, DISP SC, MYCHK…など)も結構ありますが、一番わかりにくいのは何と云ってもスクロールに関するルーチン(SCROLL, DISPLL, DISPLS)です。しかし、これが List 6-2 の花ですから、ここを理解しないで先に進むわけにはいきません。説明が最後になっているのは、《一番おいしい料理は最後に食べよう》という筆者の貧乏根性のためではなく、やっとなね合わせという前菜から、メイン・ディッシュに入ったというのが正解です。そこで、スクロール本体に入る前に、本日のメニューにあたるゲーム全体のプログラム構成を、とりあえず確認しておくことにしましょう。

右のアドレス・マップを見ると、裏 RAM もデータ・エリアとして使用することに

図 4



なっています。そのため、最終的にプログラムを動かすには、フル RAM・モードにしなければなりません。List 6-2 ではメイン RAM のまま、適当なデータでテストしていきます。もちろん、ここで正しいマップ・データや敵データを使用しないといっても、プログラムだけは正式のデータを読んでいるように作っておかなければならないのは当然のことです。そこで、スクロール・ルーチンを理解する前に、マップ・データの構造がどうなっているか、確認する必要があります。

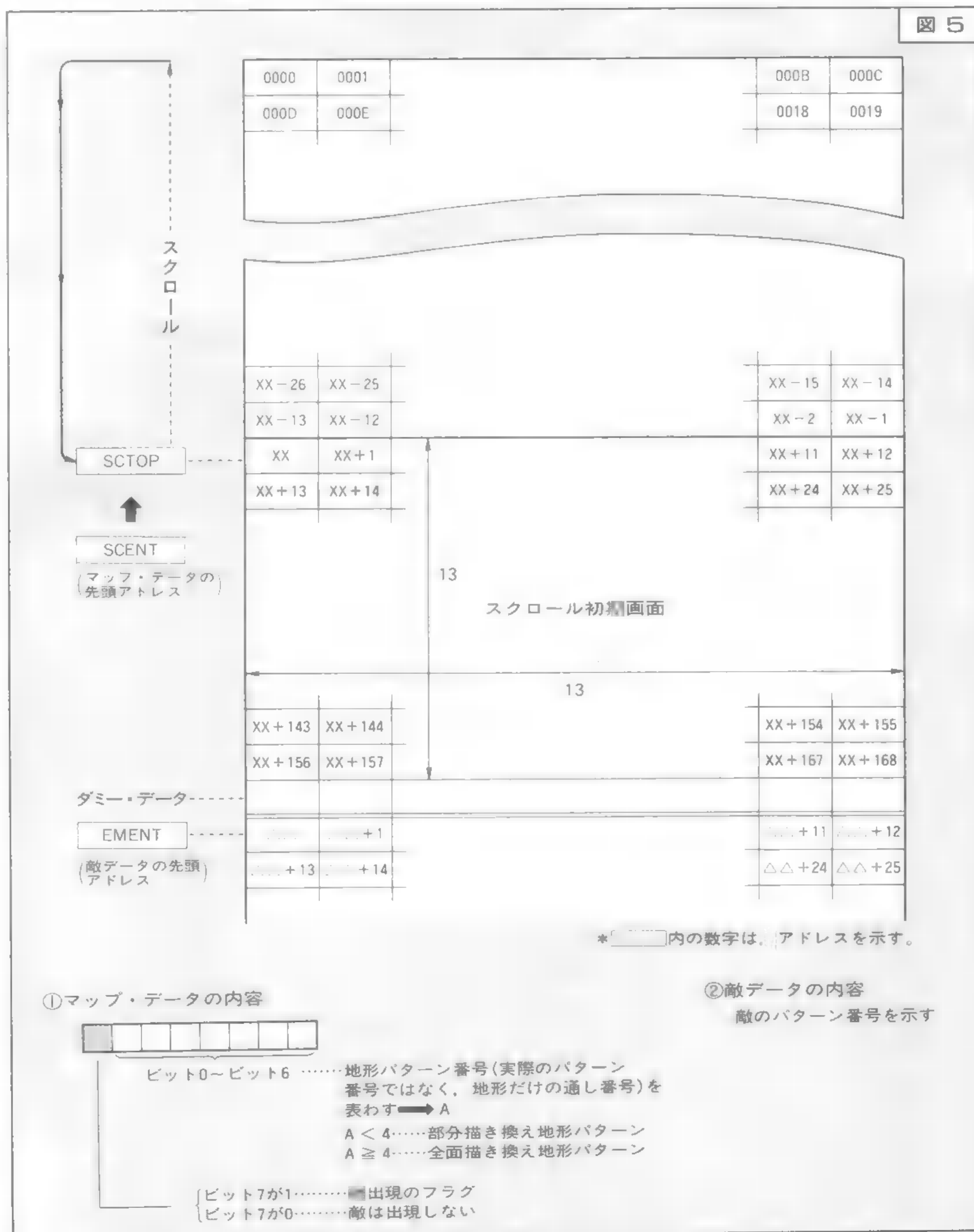
地形パターンのサイズは、他のパターンと同じように 32×16 ドットで、スクロール画面にはそのパターンが横 13 個、縦 12.5 個表示されることになります。データとしては端数は関係ありませんから、13×13 個のデータが必要となります。そして、スクロール開始時には SCENT で示されるアドレスが SCTOP となり、次ページの図 5 の「スクロール初期画面」にあるように、アドレスの大きい方に向かってデータを読み、画面に表示していくのです。SCTOP は、スクロールするにつれて、1 段ずつ 0000H 地形に近づいていくことになります。しかし、実際のスクロールは 2 ドット単位ですから、画面最上段に表示されるパターンは、最初 16 ドット全部ではなく、下 2 ドット分だけということになります。したがって、画面背景上段にパターン全部が表示されるのは、スクロールを 8 回繰り返した後ということです。

この最上段に表示するパターンのドット数を決めるため、7~0 を繰り返すスクロール・カウンター(SCCT)を用意しています。

また、このスクロール・カウンターの値が 0 になった時、すなわち最上段に地形パターン全部が表示される時には、地形を表示する直前に、敵が出現するかどうかのチェックを行ない、マップ・データのビット 7 が 1 であれば、敵出現ルーチンをコールするようになっています。敵出現のプログラムは、次の List 6-3 に書かれているため、ここでは何もせずにリターンするようになっていますが、実際には敵の表示および初期データの設定が行われることになります。こうして、最上段のパターンが全部表示されると、次回からは SCTOP が次のデータ・アドレス(SCTOP+13)に進んでいく…ということを、繰り返しているのです。

スクロールについてのもう 1 つの特徴は、地形パターンを「部分描き換えパターン」と「全面描き換えパターン」とに分けていることです。「全面描き換えパターン」というのは、文字通りスクロールするたびに、パターン全部を描き直さなければならない通常のパターンのことですが、「部分描き換えパターン」の方は必要があれば描くという、手抜きのパターンのことなのです。具体的にいうと、縦 2 ドット単位で同じパターンを並べて作った地形パターンということになります。これは、下の段に同じパターンが描かれていれば、スクロールによる描き換えはまったく不要ということを意味します。また、下の段がたとえ違うパターンであっても、描くのは下の 2 ドット分だけですみますから、スクロールの速度アップには大いに貢献してくれるのです。図 5 で、ダミー・データとして 13 バイト確保しているのは、この下の段チェックのためな

図 5



のです。

この「部分描き換えパターン」というのは、実際にはそれほど多くのバリエーションがありませんから、ここではとりあえず4種類(地形パターン番号0~3)しか用意しませんでした。しかし、地形パターンとしては全部で52種類のパターンが作れるようにしていますので、希望により全面描き換えパターンの先頭番号(LPL1)の値を換えれば、その数は増やすことができます。

なお、このスクロール・ゲームのテスト実

行アドレスは、これまでと違いすべてC800H番地からなっていますので、間違えないようにしてください。また、今回メモリーにロードしたプログラムの内、セーブしておかなければならないのはBD00H~C1CEHの部分です。パターン・データがなくても、テストに支障はありませんが、マップ・データは適当なので、長くやっているとワークエリアの値を読むこともあります。その場合には、画面が狂うことも考えられますが、ここでは気にせずにテストをしてください。

List 6-2 スクロール・ゲームと重ね合わせ処理

アセンブルした後セーブしておくこと

```

***** List 6-2 *****
;
C000      VTOP: EQU 0C000H ;V ram 1F address
0050      HLEN: EQU 80    ;Horizontal LENGt
0006      WTIMES: EQU 6    ;Waiting TIMES
;
;          ORG 0BD00H
;
BD00      INTTBL: ;Interrupt Table
BD00 00001080    DW 0,VTCT,0,0
BD04 00000000
BD08 00000000    DW 0,0,0,0
BD0C 00000000
;
BD10      VTCT: ;Vertical Frame Counter
BD10 F3          DI
BD11 08          EX AF,AF
BD12 09          EXX
BD13 211FBD      LD HL,COUNT
BD16 34          INC (HL)
BD17 3E02        LD A,2
BD19 D3E4        OUT (0E4H),A
BD1B 09          EXX
BD1C 08          EX AF,AF
BD1D FB          EI
BD1E C9          RET
;
BD1F      COUNT: ;COUNTER
BD1F            DS 1
;
BD20      WAIT: ;Waiting
BD20 3A1FBD      LD A,(COUNT)
BD23 FE06        CP WTIMES
1310 BD25 38F9      JR C,WAIT

```

```

BD27 AF          XOR A
BD28 321FBD      LD (COUNT),A
BD2B C9          RET

;
SETINT: ;SEI Interrupt
BD2C F3          DI
BD2C 3EB0        LD A,0BDH
BD2F ED47        LD I,A
BD31 AF          XOR A
BD32 321FBD      LD (COUNT),A
BD35 3E02        LD A,2
BD37 D3E6        OUT (0E6H),A
BD39 D3E4        OUT (0E4H),A
BD3B DB32        IN A,(32H)
BD3D F680        OR 80H
BD3F D332        OUT (32H),A
BD41 FB          EI
BD42 C9          RET

;
ORIINT: ;Original Interrupt
BD43 F3          DI
BD44 3EF3        LD A,0F3H
BD46 ED47        LD I,A
BD48 3AC3E6      LD A,(0E6C3H)
BD4B D3E4        OUT (0E4H),A
BD4D 3A0EEF      LD A,(0EF0EH)
BD50 D3E6        OUT (0E6H),A
BD52 FB          EI
BD53 C9          RET

;
DISP: ;DISPlay
BD54 COEDBE      CALL XYADR
BD57 CD04BF      CALL P0ADR
BD5A D35D        OUT (5DH),A
BD5C CD67BD      CALL BOX
BD5F D35E        OUT (5EH),A
BD61 CD67BD      CALL BOX
BD64 D35F        OUT (5FH),A
BD66 C9          RET

;
BOX: ;BOX
BD67 0610        LD B,10H
BD69 YFBOX: ;Y size free BOX
BD69 0EFF        LD C,0FFH
BD6B ED5B81BD    LD DE,(DISPAD)
BD6F LOOP: ;LOOP
BD6F EDA0        LDI
BD71 EDA0        LDI
BD73 EDA0        LDI
BD75 EDA0        LDI
BD77 7B          LD A,E
BD78 C64C        ADD A,HLEN-4
BD7A 5F          LD E,A
BD7B 3001        JR NC,SAMED
BD7D 14          INC D
BD7E SAME D: ;SAME D register
BD7E 10EF        DJNZ LOOP
BD80 C9          RET

;
DISPAD: ;DISPlay Address
BD81 DS 2
;

```



```

BD83
BD83 22AE8D
BD86 CEEDBE
BD89 AF
BD8A D35D
BD8C CD97BD
BD8F D35E
BD91 CD97BD
BD94 D35F
BD96 C9

CLPTXY:
LD (SIZE),HL
CALL XYADR
XOR A
OUT (5DH),A
CALL ERBOX
OUT (5EH),A
CALL ERBOX
OUT (5FH),A
RET

;
ERBOX:
LD HL,(DISPAD)
LD DE,HLEN
LD BC,(SIZE)

ERL1:
PUSH BC
PUSH HL

ERL2:
LD (HL),A
INC HL
DJNZ ERL2
POP HL
ADD HL,DE
POP BC
DEC C
JR NZ,ERL1
RET

;
SIZE:
DS 2

;
BLPUT:
CALL XYADR
LD DE,BDATA
LD BC,400H+HLEN

BLPL:
OUT (5DH),A
LD A,(DE)
LD (HL),A
INC DE
OUT (5EH),A
LD A,(DE)
LD (HL),A
INC DE
LD A,L
ADD A,C
LD L,A
JR NC,SAMEH
INC H

SAMEH:
DJNZ BLPL
OUT (5FH),A
RET

;
DISPLE:
CALL XYADR
CALL SEEKLO
OUT (5DH),A
CALL BOXL
LD HL,(LOADR)

```

```

BDDC D35E      OUT (5EH),A
BDDE CDE4BD    CALL BOXL
BDE1 D35F      OUT (5FH),A
BDE3 C9        RET

;
BOXL:          ;BOXL: " "
LD DE,(DISPAD)
LD BC,8FFH
LLOOP:         ;Letter
LDI
LDI
LD A,E
ADD A,HLEN-2
LD E,A
JR NC,SAMEDL
INC D
SAMEDL:        ;SAMEDL: " "
DJNZ LLOOP
RET

;
LOADR:         ;LOADR: " "
DS 2

;
CLS:           ;CLS: " "
LD A,(0E6C2H)
AND 0F7H
OUT (31H),A
OUT (5CH),A
CALL ACLS
OUT (5DH),A
CALL ACLS
OUT (5EH),A
CALL ACLS
OUT (5FH),A
LD A,(0E6C2H)
OUT (31H),A
RET
ACLS:          ;ACLS: " "
LD HL,VTOP
LD DE,VTOP+1
LD BC,3E7FH
LD (HL),0
LDIR
RET

;
DISPLL:        ;DISPLL: " "
PUSH HL
PUSH BC
INC B
DEC E
JP P,YPLUS
LD B,0
CALL XYADR
ADD A,LPS1
CALL PQADR
POP BC
PUSH BC
XOR A
SUB A
ADD A,A
ADD A,A

```

BE3E 87	ADD A,A
BE3F 5F	LD E,A
BE40 1600	LD D,0
BE42 19	ADD HL,DE
BE43 3E08	LD A,8
BE45 80	ADD A,B
BE46 87	ADD A,A
BE47 47	LD B,A
BE48 D35C	OUT (5CH),A
BE4A CD69BD	CALL YFBOX
BE4D D35F	OUT (5FH),A
BE4F C1	POP BC
BE50 E1	POP HL
BE51 C9	RET

BE52	YPLUS: ;Y=PLUS
BE52 57	LD D,A
BE53 3E5C	LD A,DEND
BE55 B8	CP B
BE56 7A	LD A,D
BE57 3812	JR C,ODEND
BE59 CDEDBE	CALL XYADR
BE5C C628	ADD A,LPS1
BE5E CD04BF	CALL PDADR
BE61 D35C	OUT (5CH),A
BE63 CD67BD	CALL BOX
BE66 D35F	OUT (5FH),A
BE68 C1	POP BC
BE69 E1	POP HL
BE6A C9	RET

BE6B	ODEND: ;Over Down END
BE6B CDEDBE	CALL XYADR
BE6E C628	ADD A,LPS1
BE70 CD04BF	CALL PDADR
BE73 C1	POP BC
BE74 C5	PUSH BC
BE75 3E64	LD A,DEND+8
BE77 90	SUB B
BE78 87	ADD A,A
BE79 47	LD B,A
BE7A D35C	OUT (5CH),A
BE7C CD69BD	CALL YFBOX
BE7F D35F	OUT (5FH),A
BE81 C1	POP BC
BE82 E1	POP HL
BE83 C9	RET

BE84	DISPL5: ;Display Land Small
BE84 57	LD D,A
BE85 3E5C	LD A,DEND
BE87 B8	CP B
BE88 F8	RET M
BE89 E5	PUSH HL
BE8A C5	PUSH BC
BE8B 78	LD A,B
BE8C C607	ADD A,7
BE8E 47	LD B,A
BE8F 7A	LD A,D
BE90 C628	ADD A,LPS1
BE92 CDEDBE	CALL XYADR
BE95 CD04BF	CALL PDADR


```

BE98 0602      LD      B,2
BE9A D35C      OUT     (5CH),A
BE9C CD69BD    CALL    YFBOX
BE9F D35F      OUT     (5FH),A
BEA1 C1        POP     BC
BEA2 E1        POP     HL
BEA3 C9        RET

;
BEA4           MAKEESC: ;MAKE screen
BEA4 D35E      OUT     (5EH),A
BEA6 0EC8      LD      C,200
BEA8 2134C0    LD      HL,0C034H
BEAB 113400    LD      DE,34H
BEAE           MSLP1: ;MakeSc Loop 1
BEAE 061C      LD      B,28
BEB0           MSLP2: ;MakeSc Loop 2
BEB0 36FF      LD      (HL),0FFH
BEB2 23        INC     HL
BEB3 10FB      DJNZ    MSLP2
BEB5 19        ADD     HL,DE
BEB6 0D        DEC     C
BEB7 20F5      JR      NZ,MSLP1

;
BEB9 D35D      OUT     (5DH),A
BEBB 0E4C      LD      C,76
BEBD 2136C0    LD      HL,0C036H
BEC0 113600    LD      DE,36H
BEC3           MSLP3: ;MakeSc Loop 3
BEC3 061A      LD      B,26
BEC5           MSLP4: ;MakeSc Loop 4
BEC5 36FF      LD      (HL),0FFH
BEC7 23        INC     HL
BEC8 10FB      DJNZ    MSLP4
BECA 19        ADD     HL,DE
BECB 0D        DEC     C
BECC 20F5      JR      NZ,MSLP3

;
BECE CDD9BE    CALL    DBOX
BED1 D35E      OUT     (5EH),A
BED3 CDD9BE    CALL    DBOX
BED6 D35F      OUT     (5FH),A
BED8 C9        RET
DBOX: ;Delete
BED9 0E44      LD      C,68
BEDB 2177C1    LD      HL,0C177H
BEDE 113800    LD      DE,38H
BEE1           DBLP1: ;DB Loop 1
BEE1 0618      LD      B,24
BEE3           DBLP2: ;DB Loop 2
BEE3 3600      LD      (HL),0
BEE5 23        INC     HL
BEE6 10FB      DJNZ    DBLP2
BEE8 19        ADD     HL,DE
BEE9 0D        DEC     C
BEEA 20F5      JR      NZ,DBLP1
BEEC C9        RET

;
XYADR: ;XY TO Address
BEED          LD      L,B
BEEF          LD      H,0
BEF0          LD      E,L
BEF1          LD      D,H

```

スクロール画面以外の画面表示

```

BEF2 29      ADD HL,HL
BEF3 29      ADD HL,HL
BEF4 19      ADD HL,DE
BEF5 29      ADD HL,HL
BEF6 29      ADD HL,HL
BEF7 29      ADD HL,HL
BEF8 29      ADD HL,HL
BEF9 29      ADD HL,HL
BEFA 59      LD E,C
BEFB 19      ADD HL,DE
BEFC 1100C0  LD DE,VTOP
BEFF 19      ADD HL,DE
BF00 2281BD  LD (DISPAD),HL
BF03 C9      RET

```

```

;
; POADR: ;Pattern Data Address
BF04      LD L,A
BF04 6F      LD H,0
BF05 2600    ADD HL,HL
BF07 29      LD DE,PDBASE
BF08 116CCB  ADD HL,DE
BF0B 19      LD A,(HL)
BF0C 7E      INC HL
BF0D 23      LD H,(HL)
BF0E 66      LD L,A
BF0F 6F      RET
BF10 C9

```

```

;
; LBASE: EQU 0BA00H ;Letter BASE

```

```

;
; SEEKLD: ;Seek Letter Data
BF11      ADD A,A
BF11 87      ADD A,A
BF12 87      LD L,A
BF13 6F      LD H,0
BF14 2600    ADD HL,HL
BF16 29      ADD HL,HL
BF17 29      LD DE,LBASE
BF18 1100BA  ADD HL,DE
BF1B 19      LD (LOADR),HL
BF1C 22F9BD  RET
BF1F C9

```

```

;
; MSGPRN: ;MeSSaGe PRiNt
BF20      LD A,(HL)
BF20 7E      OR A
BF21 B7      RET Z
BF22 C8      CP ' '
BF23 FE20    JR NZ,MSG2
BF25 2002    LD A,'0'+10
BF27 3E3A    MSG2: ;MSGprn 2
BF29      SUB '0'
BF29 D630    CP 11
BF2B FE08    JR C,MSG1
BF2D 3802    SUB 6
BF2F D606    MSG1: ;MSGprn 1
BF31      PUSH BC
BF31 C5      PUSH HL
BF32 E5      CALL DISPLE
BF33 CDCEBD  POP HL
BF36 E1      POP BC
BF37 C1      INC C
BF38 0C      INC C
BF39 0C      INC HL
BF3A 23      JR MSGPRN
BF3B 18E3

```

```

0E3B      ; SCLOC: EQU 0E3BH ;Score LOcation
;
BF3D      ;DISPSC: ;DISPlay SCore
BF3D 01380E LD BC,SCLOC
BF40 2171BF LD HL,SCOREL
BF43 7B LD A,E
BF44 86 ADD A,(HL)
BF45 27 DAA
BF46 77 LD (HL),A
BF47 2B DEC HL
BF48 7A LD A,D
BF49 8E ADC A,(HL)
BF4A 27 DAA
BF4B 77 LD (HL),A
BF4C 2B DEC HL
BF4D 3E00 LD A,0
BF4F 8E ADC A,(HL)
BF50 27 DAA
BF51 77 LD (HL),A
BF52 CD5ABF CALL SCOREP
BF55 23 INC HL
BF56 CD5ABF CALL SCOREP
BF59 23 INC HL

; SCOREP: ;SCORE Print
BF5A      LD A,(HL)
BF5A 7E RLCA
BF5B 07 RLCA
BF5C 07 RLCA
BF5D 07 RLCA
BF5E 07 RLCA
BF5F CD63BF CALL PRINTF
BF62 7E LD A,(HL)

; PRINTF: ;PRINT Figure
BF63      AND 0FH
BF63 E60F PUSH BC
BF65 C5 PUSH HL
BF66 E5 CALL DISPLE
BF67 CDCEBD POP HL
BF6A E1 POP BC
BF6B C1 INC C
BF6C 0C INC C
BF6D 0C INC C
BF6E C9 RET

; SCORE2: ;SCORE 2
BF6F      DS 1
BF6F      SCORE1: ;SCORE 1
BF70      DS 1
BF70      SCOREL: ;SCORE Low
BF71      DS 1
BF71      DUMMY: ;score DUMMY
BF72      DB '00',0
BF72 303000

; RND: ;RaNDom figure
BF75      PUSH HL
BF75 E5 LD HL,(RNDWOK)
BF76 2A88BF LD D,H
BF79 54 LD E,L
BF7A 5D ADD HL,HL
BF7B 29 ADD HL,HL
BF7C 29 ADD HL,HL
BF7D 19 ADD HL,DE
BF7E 117335 LD DE,3573H

```

現在
加算

表示される得点を


```

4660 BF81 19          ADD HL,DE
      BF82 2288BF    LD (RNDWOK),HL
      BF85 7C        LD A,H
      BF86 E1        POP HL
      BF87 C9        RET

      ;
      RNDWOK: ;RaNDom figure Work area
      BF88 711F      DB 113,31

      ;
      MYMOVE: ;MY MOVement
      BF8A ED48D0CE   LD BC,(MYLOC)
      BF8E DB01       IN A,(1)
      BF90 1F         RRA
      BF91 DB00       IN A,(0)
      BF93 3800       JR C,NOT8
      BF95 E650       AND 50H
      BF97 EA66C0     JP PE,KEY8
      BF9A E640       AND 40H
      BF9C CA7DC0     JP Z,KEY68
      BF9F C33EC0     JP KEY48

      ;
      NOT8: ;NOT pressed key 8
      BFA2 CB57       BIT 2,A
      BFA4 200B       JR NZ,NOT28
      BFA6 E650       AND 50H
      BFA8 EADF8F     JP PE,KEY2
      BFAB E640       AND 40H
      BFAD 2844       JR Z,KEY26
      BFAF 180B       JR KEY24

      ;
      NOT28: ;NOT pressed key 2 nor 8
      BFB1 E650       AND 50H
      BFB3 EAA4C0     JP PE,MDISP
      BFB6 E640       AND 40H
      BFB8 2870       JR Z,KEY6
      BFBA 1858       JR KEY4

      ;
      KEY24: ;pressed KEY = 2 + 4
      BFBC 3E5C       LD A,DEND
      BFBE B8         CP 8
      BFBF 2853       JR Z,KEY4
      BFC1 AF         XOR A
      BFC2 B9         CP C
      BFC3 2820       JR Z,K20K
      BFC5 210204     LD HL,402H
      BFC8 CD83BD     CALL CLPTXY
      BFCB ED48D0CE   LD BC,(MYLOC)
      BFCF 04         INC B
      BFD0 C5         PUSH BC
      BFD1 0C         INC C
      BFD2 0C         INC C
      BFD3 0C         INC C
      BFD4 210E01     LD HL,10EH
      BFD7 CD83BD     CALL CLPTXY
      BFDA C1         POP BC
      BFDB 0D         DEC C
      BFDC C3A0C0     JP MMDISP

      ;
      KEY2: ;pressed KEY = 2
      BDF0 3E5C       LD A,DEND
      BDF1 B8         CP 8
      BFE1 0D         DEC C
      BFE2 CAA4C0     JP Z,MDISP

```

BFE5		K2OK:	;Key 2 direction OK	
BFE5	210204	LD	HL,402H	移動方向=2の不要部分消去+次座標計算
BFE8	CD83BD	CALL	CLPTXY	
BFE8	ED4BD0CE	LD	BC,(MYLOC)	
BFEF	04	INC	B	
BFF0	C3A0C0	JP	MMDISP	
BFF3		;		
BFF3	3E30	KEY26:	;pressed KEY = 2 + 6	
BFF5	B9	LD	A,REND	
BFF6	28E7	CP	C	
BFF8	3E5C	JR	Z,KEY2	
BFFA	B8	LD	A,DEND	
BFFB	2833	CP	B	
BFFD	210204	JR	Z,K6OK	
C000	CD83BD	LD	HL,402H	移動方向=6の不要部分消去+次座標計算
C003	ED4BD0CE	CALL	CLPTXY	
C007	04	LD	BC,(MYLOC)	
C008	C5	INC	B	
C009	210E01	PUSH	BC	
C00C	CD83BD	LD	HL,10EH	
C00F	C1	CALL	CLPTXY	
C010	0C	POP	BC	
C011	C3A0C0	INC	C	+1,B+1
		JP	MMDISP	MMDISP
C014		;		
C014	AF	KEY4:	;pressed KEY = 4	
C015	B9	XOR	A	
C016	CAA4C0	CP	C	
C019	0C	JP	Z,MDISP	
C01A	0C	K4OK:	;Key 4 direction OK	移動方向=4の不要部分消去+次座標計算
C01B	0C	INC	C	
C01C	211001	INC	C	
C01F	CD83BD	INC	C	
C022	ED4BD0CE	LD	HL,110H	
C026	0D	CALL	CLPTXY	
C027	C3A0C0	LD	BC,(MYLOC)	
		DEC	C	(B) * (C-1,B)
		JP	MMDISP	MMDISP
C02A		;		
C02A	3E30	KEY6:	;pressed KEY = 6	
C02C	B9	LD	A,REND	
C02D	CAA4C0	CP	C	右エント)なら MDISPへ
C030	211001	JP	Z,MDISP	
C033	CD83BD	K6OK:	;Key 6 direction OK	移動方向=6の不要部分消去+次座標計算
C036	ED4BD0CE	LD	HL,110H	
C03A	0C	CALL	CLPTXY	
C03B	C3A0C0	LD	BC,(MYLOC)	
		INC	C	(B) * (C+1,B)
		JP	MMDISP	MMDISP
C03E		;		
C03E	AF	KEY48:	;pressed KEY = 4 + 8	
C03F	B9	XOR	A	
C040	2824	CP	C	
C042	3C	JR	Z,KEY8	
C043	B8	INC	A	
C044	30D3	CP	B	B≦1(上エント)なら K4OKへ
C046	0C	JR	NC,K4OK	
C047	0C	INC	C	移動方向=8の不要部分消去+次座標計算
C048	0C	INC	C	
C049	210C01	INC	C	
C04C	CD83BD	LD	HL,10CH	
		CALL	CLPTXY	

```

C04F ED4BD0CE      LD      BC,(MYLOC)
C053 78             LD      A,B
C054 C606           ADD     A,6
C056 47             LD      B,A
C057 210404         LD      HL,404H
C05A CD83BD         CALL    CLPTXY
C05D ED4BD0CE      LD      BC,(MYLOC)
C061 05             DEC     B
C062 05             DEC     B
C063 0D             DEC     C
C064 183A           JR      MMDISP

;
KEY8: ;pressed KEY = 8
C066               LD      A,1
C066 3E01           CP      B
C068 B8             JR      NC,MDISP
C069 3039           JR      NC,MDISP
C06B               ;Key 8 direction UP
K8OK: LD      A,B
C06B 78             ADD     A,6
C06C C606           LD      B,A
C06E 47             LD      HL,404H
C06F 210404         CALL    CLPTXY
C072 CD83BD         LD      BC,(MYLOC)
C075 ED4BD0CE      DEC     B
C079 05             DEC     B
C07A 05             JR      MMDISP
C07B 1823           ;
KEY68: ;pressed KEY = 6 + 8
C07D               LD      A,1
C07D 3E01           CP      B
C07F B8             JR      NC,KEY6
C080 30A8           LD      A,REND
C082 3E30           CP      C
C084 B9             JR      Z,K8OK
C085 28E4           LD      HL,110H
C087 211001         CALL    CLPTXY
C08A CD83BD         LD      BC,(MYLOC)
C08D ED4BD0CE      INC     C
C091 0C             PUSH    BC
C092 C5             LD      A,B
C093 78             ADD     A,6
C094 C606           LD      B,A
C096 07             LD      HL,304H
C097 210403         CALL    CLPTXY
C09A CD83BD         POP     BC
C09D C1             DEC     B
C09E 05             DEC     B
C09F 05             ;
MMDISP: ;My Move & DISPlay
C0A0 ED43D0CE      LD      (MYLOC),BC
C0A4               ;My DISPlay
MDISP: XOR     A
C0A4 AF             CALL    DISP
C0A5 CD54BD         RET
C0A8 C9             ;
MYCHK: ;MY Check
C0A9 ED58D0CE      LD      DE,(MYLOC)
C0A9 210CCE         LD      HL,EMBWK
C0AD 0628           LD      B,EMBVAL
C0B2               ;EnemY Bullet Check Loop
EMBCLP: LD      A,(HL)
C0B2 7E             OR      A
C0B3 B7             JR      Z,IHL4
C0B4 280F           ;

```

-----自機と敵の弾、敵との衝突チェック
 -----自機と敵の弾、敵との衝突チェック
 -----自機と敵の弾、敵との衝突チェック

-----自機と敵の弾、敵との衝突チェック

C0B6 23	INC HL	
C0B7 7E	LD A,(HL)	
C0B8 93	SUB E	敵の座標 - 自機座標
C0B9 FE04	CP 4	1/255
C0BB 3009	JR NC,IHL3	
C0BD 23	INC HL	
C0BE 7E	LD A,(HL)	
C0BF 92	SUB D	敵の座標 - 自機座標
C0C0 FE07	CP 7	座標が7よりIHL2
C0C2 3003	JR NC,IHL2	
C0C4 C9	RET	1/255
C0C5	IHL4: ;Inc HL 4 times	
C0C5 23	INC HL	
C0C6	IHL3: ;Inc HL 3 times	
C0C6 23	INC HL	
C0C7	IHL2: ;Inc HL 2 times	
C0C7 23	INC HL	
C0C8 23	INC HL	
C0C9 10E7	DJNZ EMBCLP	
	;	
C0CB 212CCC	LD HL,EMWORK	HL: 敵のワークエリア
C0CE 061E	LD B,EMVAL	B: 敵の座標
C0D0	WECLP: ;With Enemy Check Loop	p 213 参照
C0D0 C5	PUSH BC	
C0D1 7E	LD A,(HL)	
C0D2 3C	INC A	敵の座標 + 1
C0D3 201C	JR NZ,NEXTE	なければ NEXT
C0D5 44	LD B,H	
C0D6 4D	LD C,L	
C0D7 03	INC BC	敵の座標 + 1
C0D8 0A	LD A,(BC)	敵の座標 + 1
C0D9 FE04	CP SKYP1	敵の座標 + 1
C0DB 3814	JR C,NEXTE	座標が14よりIHL2
C0DD 03	INC BC	
C0DE 0A	LD A,(BC)	
C0DF 93	SUB E	敵の座標 - 自機の座標
C0E0 C602	ADD A,2	座標 + 2
C0E2 FE05	CP 5	座標が5よりIHL2
C0E4 300B	JR NC,NEXTE	
C0E6 03	INC BC	
C0E7 0A	LD A,(BC)	
C0E8 92	SUB D	敵の座標 - 自機の座標
C0E9 C605	ADD A,5	座標 + 5
C0EB FE0B	CP 0BH	座標が0BHよりIHL2
C0ED 3002	JR NC,NEXTE	
C0EF C1	POP BC	
C0F0 C9	RET	衝突の座標 - キャリーフラグを伴ってリターン
C0F1	NEXTE: ;NEXT Enemy	
C0F1 011000	LD BC,EMWLEN	HL: 敵のワークエリア
C0F4 09	ADD HL,BC	
C0F5 C1	POP BC	
C0F6 10D8	DJNZ WECLP	
C0F8 B7	OR A	
C0F9 C9	RET	
	;	
C0FA	SSKCK: ;Space & Shift Key Check	
C0FA DB08	IN A,(8)	
C0FC 4F	LD C,A	
C0FD DB09	IN A,(9)	
C0FF A1	AND C	
C100 E640	AND 40H	座標が40HよりIHL2
C102 21D3CE	LD HL,SSKEY	
C105 2803	JR Z,SSP	押されていない場合は(HL)・FFHをし、リターン

17160	C107 36FF	LD (HL),0FFH	
	C109 C9	RET	
	C10A	SSP: ;Space or Shift is Pressed	
	C10A 34	INC (HL)	HL ← (HL)+1
	C10B C0	RET NZ	HL: 弾のワークエリア・先頭アドレス
	C10C 36F8	LD (HL),0F8H	B: 弾の長さ
	C10E 060C	LD B,MYBVAL	H: 弾のワークエリア・先頭アドレス
	C110 21ACCE	LD HL,MYBWOK	L: 弾のワークエリアの長さ
	C113 110300	LD DE,MYBWLE	D: 弾のワークエリアの長さ
	C116 0E00	LD C,0	弾のワークエリアの長さ
	C118 CD20C1	CALL BWCK	ワークエリアの長さ
	C11B D8	RET C	
	C11C 05	DEC B	
	C11D C8	RET Z	
	C11E 0E03	LD C,3	
	C120	BWCK: ;Bullet Work area Check	
	C120 7E	LD A,(HL)	
	C121 B7	OR A	
	C122 2805	JR Z,NBOK	
	C124 19	ADD HL,DE	
	C125 10F9	DJNZ BWCK	
	C127 37	SCF	
	C128 C9	RET	
	C129	NBOK: ;New Bullet OK	
	C129 3601	LD (HL),1	
	C12B 23	INC HL	
	C12C 3AD0CE	LD A,(MYLOC)	
	C12F 81	ADD A,C	
	C130 77	LD (HL),A	
	C131 23	INC HL	
	C132 3AD1CE	LD A,(MYLOC+1)	
	C135 77	LD (HL),A	
	C136 23	INC HL	
	C137 C9	RET	
	C138	MYBMOV: ;MY Bullet MOVE	
	C138 21ACCE	LD HL,MYBWOK	
	C13B 060C	LD B,MYBVAL	
	C13D	MYBLP: ;MY Bullet Loop	
	C13D 7E	LD A,(HL)	
	C13E B7	OR A	
	C13F 2826	JR Z,NEXTMB	
	C141 C5	PUSH BC	
	C142 23	INC HL	
	C143 4E	LD C,(HL)	
	C144 23	INC HL	
	C145 46	LD B,(HL)	
	C146 E5	PUSH HL	
	C147 C5	PUSH BC	
	C148 210401	LD HL,104H	
	C14B CD83BD	CALL CLPTXY	
	C14E C1	POP BC	
	C14F 78	LD A,B	
	C150 D602	SUB 2	
	C152 47	LD B,A	
	C153 3008	JR NC,MBPUT	
	C155 E1	POP HL	
	C156 2B	DEC HL	
	C157 2B	DEC HL	
	C158 3600	LD (HL),0	
17760	C15A C1	POP BC	

17770	C15B 180A	JR NEXTMB	NEXTMB	
	C150	MBPUT: ;My Bullet PUT		
	C150 C5	PUSH BC		
	C15E CDB0BD	CALL BLPUT		C.5に弾の表示
	C161 C1	POP BC		
	C162 E1	POP HL		
	C163 70	LD (HL),B		(自機の弾のY座標)+B
	C164 C1	POP BC		
	C165 1802	JR NMB2		
	C167	NEXTMB: ;NEXT My Bullet		
	C167 23	INC HL		
	C168 23	INC HL		
	C169	NMB2: ;NextMB 2		HL: 次の自機の弾のワークエリア
	C169 23	INC HL		
	C16A 1001	DJNZ MYBLP		
	C16C C9	RET		
		; SCROLL: ;SCROLL		
	C160	LD HL,SCCT		スクロールカウンターの値を-1する
	C160 2108CE	DEC (HL)		
	C170 35	LD A,(HL)		
	C171 7E	LD A,(HL)		
	C172 E607	AND 7		A=7~0のループとなる
	C174 2AD6CE	LD HL,(SCTOP)		現在画面のマッパ・データ先頭アドレス
	C177 2016	JR NZ,SDSCR		A≠0ならSDSCRへジャンプ(7ビット)
	C179 EB	EX DE,HL		DE←HL
	C17A 21F3FF	LD HL,-13		HL←HL-13 マッパ・データを進める
	C17D ED5A	ADC HL,DE		(ADD HL,DEではゼロフラグが変化しない)
	C17F 200A	JR NZ,DCONT		HL≠0ならDCONTへ
	C181 2ADBCE	LD HL,(EMENT)		
	C184 2B	DEC HL		敵データ・ポインタを初期化
	C185 22D4CE	LD (EDPO),HL		
	C188 2AD9CE	LD HL,(SCENT)		HL←マッパ・データのスタート・アドレス
	C188	DCONT: ;Data Continue		
	C188 22D6CE	LD (SCTOP),HL		次の画面のマッパ・データ先頭アドレス
	C18E EB	EX DE,HL		HL←現在画面のマッパ・データ先頭アドレス
		; SDSCR: ;Same Data SCROLL		
	C18F	NEG		A←-A
	C18F ED44	LD B,A		B←A 画面表示スタートY座標(-7~0)
	C191 47	LD C,0		C←0 画面表示スタートX座標
	C192 0E00			
	C194	SCRLP: ;SCROLL Loop		
	C194 AF	XOR A		
	C195 B8	CP B		B≠0なら ONLYSへ敵の出現はない
	C196 2006	JR NZ,ONLYS		
	C198 7E	LD A,(HL)		HL≧80Hなら EMAPPをコールする
	C199 FE80	CP 80H		ビット7=1が敵出現のフラグ
	C19B D400C3	CALL NC,EMAPP		
	C19E	ONLYS: ;ONLY Scroll		
	C19E 7E	LD A,(HL)		A←地形パターン番号
	C19F E67F	AND 7FH		
	C1A1 FE04	CP LPL1-LPS1		A≧4ならCDLLへ全面描き換えの地形パターン
	C1A3 3013	JR NC,CDLL		
	C1A5 EB	EX DE,HL		DE←HL
	C1A6 210D00	LD HL,13		HL←HL+1
	C1A9 19	ADD HL,DE		前の画面のマッパ・データ先頭アドレス
	C1AA AE	XOR (HL)		A←前(下の段)の地形パターン番号と違うビット
	C1AB E67F	AND 7FH		か1になっている
	C1AD EB	EX DE,HL		HL←現在の画面のマッパ・データ先頭アドレス
	C1AE 280B	JR Z,NEXTLD		A=0ならNEXTLDへ(下の段と同じ)
	C1B0 7E	LD A,(HL)		A←地形パターン番号
	C1B1 E67F	AND 7FH		め描き換え不要
18370				

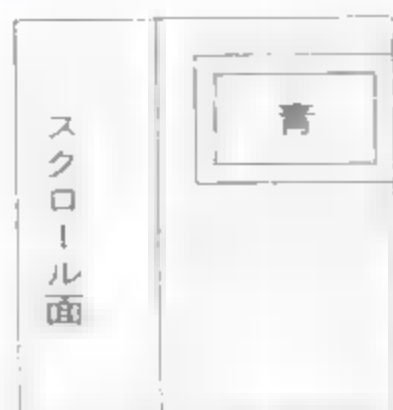
C1B3 CD84BE	CALL DISPLS	A 8 番号を表示
C1B6 1803	JR NEXTLD	
C1B8	; CDLL: ;Call DispLL	
C1B8 CD27BE	CALL DISPLL	A 全面描き除きの20# (番号) を
C1B8	; NEXTLD: ;NEXT Land	
C1BB 23	INC HL	
C1BC 79	LD A,C	
C1BD C604	ADD A,4	
C1BF 4F	LD C,A	
C1C0 FE34	CP REND+4	
C1C2 20D0	JR NZ,SCRLP	
C1C4 0E00	LD C,0	
C1C6 78	LD A,B	
C1C7 C608	ADD A,8	
C1C9 47	LD B,A	
C1CA FE64	CP DEND+8	
C1CC 38C6	JR C,SCRLP	
C1CE C9	RET	
	; ORG 0C300H	
C300	; EMAPP: ;EneMy APPEAR	
C300 C9	RET	敵出現
	; ORG 0C800H	
C800	; TEST: ;TEST	
C800 F3	DI	
C801 AF	XOR A	
C802 D351	OUT (51H),A	
C804 3100B6	LD SP,0B600H	
C807 CDFB8D	CALL CLS	画面消去
	; LD HL,(SCENT)	
C80A 2AD9CE	LD B,182	
C80D 06B6	MAPLP: ;MAPping Loop	
C80F 3604	LD (HL),4	表示
C811 23	INC HL	
C812 10FB	DJNZ MAPLP	
	; LD HL,(SCENT)	
C814 2AD9CE	LD (SCTOP),HL	
C817 22D6CE	XOR A	
C81A AF	LD (SCCT),A	
C81B 32D8CE		
	; LD HL,5B16H	
C81E 21165B	LD (MYLOC),HL	自機の初期出現位置設定
C821 22D0CE		
	; LD HL,MYBWOK	
C824 21ACCE	LD DE,MYBWLE	
C827 110300	LD B,MYBVAL	
C82A 060C	IMYBLP: ;Initialize MY Bullet Loop	自機の弾のクエリア初期化
C82C	LD (HL),0	出現フラグを、すべて0にする
C82C 3600	ADD HL,DE	
C82E 19	DJNZ IMYBLP	
C82F 10FB	CALL SETINT	割り込みモードの設定
C831 CD2CBD		
	; MAIN: ;MAIN loop	
C834	CALL SSKCK	SPACE, SHIFT のチェック
C834 CDFAC0	CALL MYBMV	自機の移動
C837 CD38C1		

```
C83A CD6DC1      CALL SCROLL      ;スクロー
C83D CD38C1      CALL MYBMOV      ;自機の弾移動
C840 CD8ABF      CALL MYMOVE      ;敵の弾移動
C843 CD20BD      CALL WAIT        ;待つ
C846             M1: ;Main 1
C846 DB08         IN  A,(8)        ;IN A,(8)
C848 1F          RRA              ;右にシフト
C849 38E9         JR   C,MAIN      ;C=1ならMAIN
C84B 1F          RRA              ;右にシフト
C84C 38F8         JR   C,M1        ;C=1ならM1
C84E CD43BD      CALL ORIINT      ;ORIINT
C851 FF          RST  38H          ;RST 38H

;
;      ORG  0CB00H

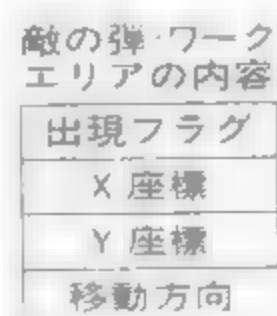
;
;EMTTBL:DS  54    ;EneM, Type TaBLe
;EMSTBL:DS  54    ;EneMy Score TaBLe
;
;SKYP1: EQU  4    ;SKY Pattern 1
;EXPP1: EQU 32    ;EXPlosion Pattern 1
;LPS1:  EQU 40    ;Land Pattern Small 1
;LPL1:  EQU 44    ;Land Pattern Large 1
;7F00   BDATA: EQU 7F00H ;Bullet DATA
;CB6C   PDBASE:DS 0C0H    ;Pattern Data BASE address
;
;EMWLEN:EQU 16    ;EneMy Work LENgth
;EMVAL: EQU 30    ;EneMy VALue
;CC2C   EMWORK:DS 480    ;EneMy WORK area
;
;EMBWLE:EQU 4     ;EneMy Bulett Work LENgth
;EMBVAL:EQU 40    ;EneMy Bulett VALue
;CE0C   EMBWOK:DS 160    ;EneMy Bulett WORK area
;
;MYBWLE:EQU 3     ;MY Bullet Work LENgth
;MYBVAL:EQU 12    ;MY Bullet VALue
;CEAC   MYBWOK:DS 36     ;MY Bullet WORK area
;
;REND:  EQU 48    ;Right END
;DEND:  EQU 92    ;Down END
;
;MYLOC: DS  2     ;MY LOcAtion
;MYRST: DS  1     ;MY ReST
;CED3   SSKEY: DS 1     ;Space & Shift KEY
;
;EDPO:  DS  2     ;Enemy Data POinter
;CED6   SCTOP: DS 2     ;SCroll TOP address
;CED8   SCCT:  DS 1     ;SCroll Counter
;
;CED9 00E0      SCENT: DW 0E000H;SCroll data ENTry
1949: CEDB 00E0      EMENT: DW 0E000H;EneMy data ENTry
```

1

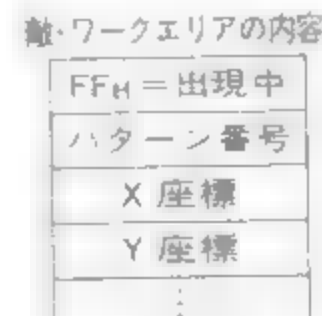


List 6-4 で使用

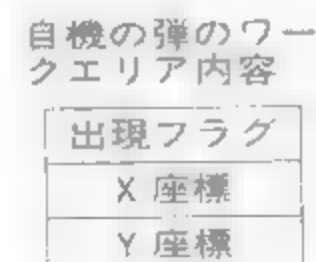
2



3



4



3. QRL…パターン・コントロール言語

スクロール・ゲームの醍醐味といえば、やはり次々と襲ってくる謎の飛行物体との、ハデな空中戦ということになりますが、ここでも重要なのはいかにして敵に生命を与えるかということです。これまでの2つのゲームにおいては、データによる移動および迷路内での追跡という形で、それぞれ一応敵らしくさせてきました。今回は、これらをさらに一歩進めた形のQRL(Quasi Robot Language=擬似ロボット言語)と名付けたコマンドで、敵を動かすことにしました。

ロボットといえば、鉄人28号と鉄腕アトムが日本代表(?)として欠かせませんが、この2つはまったく違ったタイプのロボットであるといえます。つまり、鉄人28号の方はリモコン操作により動かす、いわば操縦者の化身なのですが、アトムの方は人

工知能を持ったロボットなので、持ち主の意思とは違う行動を取ることもあるわけです。どちらも、それぞれに魅力がありますが、これをゲームに当てはめると、キー操作によって動かす主人公は鉄人タイプ、勝手に動く敵はアトム・タイプということができます。

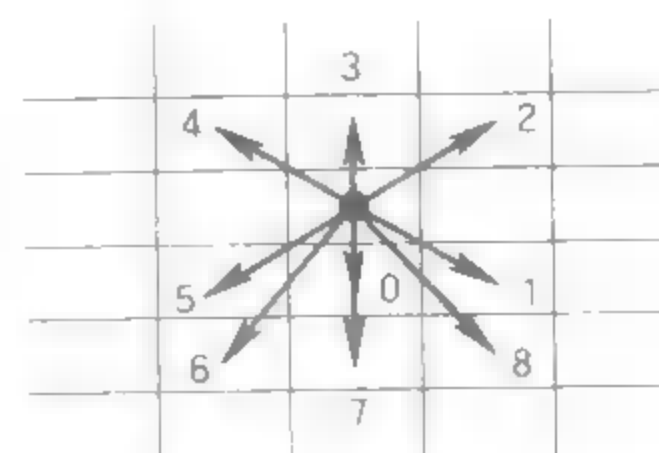
いくらアトム・タイプといっても、ここでは弾を発射することと、移動方向を決めることくらいがほとんどで、それ以上の思考力は今後の博士(あなたのこですヨ)のアイディアに期待がかかっています。この程度なら、List 3-4 で使われた移動方向データと大差ない、と思った方もいるかもしれませんが、このQRLは単なる移動方向データの集まりではなく、実行能力もあるコマンド群なのです。実は、List 3-4 にもNP(New Pointer)という、移動方向データ



表 2

QRL コマンド一覧表

	コマンド名	移動方向	弾の発射-1	弾の発射-2	弾の発射-3
			ESHOT 1	ESHOT 2	ESHOT 3
一般コマンド	STOP	00H	+ 10H	+ 20H	+ 30H
	RR	01H	+ 10H	+ 20H	+ 30H
	UR	02H	+ 10H	+ 20H	+ 30H
	UU	03H	+ 10H	+ 20H	+ 30H
	UL	04H	+ 10H	+ 20H	+ 30H
	DL	05H	+ 10H	+ 20H	+ 30H
	DD	06H	+ 10H	+ 20H	+ 30H
	DR	07H	+ 10H	+ 20H	+ 30H



移動方向

レベル	確 率
1	5/256
2	32/256
3	256/256

弾の発射レベル

	コマンド形式	コマンド	内 容
特殊コマンド	α END	80H	自爆せよ
	α JUMP <NP>	81H	<NP>へジャンプ
	α IFZ <SC> <NP>	82H	<SC>をコールしてゼロフラグが立っていれば<NP>へジャンプ, 立っていないければスキップして次のコマンドへ行け
	α IFC <SC> <NP>	83H	<SC>をコールしてキャリーフラグが立っていれば<NP>へジャンプ, 立っていないければスキップして次のコマンドへ行け
	α CALL <SC>	84H	<SC>をコールする
	α FETCH <SC>	85H	<SC>をコールし得た値(Aレジスタ)を実行する

ではないコマンドがすでに存在していたことを覚えているでしょうか。QRL とは、この移動方向以外のことをも示す、敵に対するトータル・コマンドと思えばいいのです。

では、QRL でどのようなことをさせられるのか、上の一覧表で見てください。

QRL は大きく分けると、一般コマンドと特殊コマンドから成り立っていますが、一

般コマンドでは単に移動方向を示すだけでなく、弾をレベル別(頻度別)に発射させることができます。また、敵の移動方向が自機の時と違っていますが、これは基本的な進行方向が、自機=上方向、敵=下方向と違うためです。このゲームでは、自機、敵共に移動方向の総数は停止(敵の場合は下へ1コマ移動)を含めて9方向としましたが、敵についてはさらに左右と上部への2コマ移動を加えた方がよかったかもしれません。弾の発射に関しては、QRLでは空中敵■(ESHOT1-3)と、地上敵用(LSHOT1-3)の2つを用意しています。これは発射確率と発射方向(LSHOTは自機を狙う)の違いなのですが、特殊コマンドによって空中敵でもLSHOTを利用することができます。ただ、空中敵の場合は移動方向に合わせた方が自然なので、本書で作成した空中敵(List 6-5)はESHOTだけで弾を発射しています。

次に特殊コマンドを見てみましょう。コマンド番号を、わかりやすいようにプログラムで使われているコマンド名(ラベル)で

表現すると、次のようになります。

```
80H:@END = @END  command of QRL
81H:@JUMP = @JUMP command of QRL
82H:@IFZ = @IFZ  command of QRL
83H:@IFC = @IFC  command of QRL
84H:@CALL = @CALL command of QRL
85H:@FETCH = @FETCH command of QRL
```

このコマンド自体の内容は、表2に書かれている通りで、それほど複雑なことはさせていません。しかし、たったこれだけで何でもできるだけの可能性を秘めているのです。その秘密は、コマンドの次の2バイトをサブ・コマンドとしてコールできる、という所に隠されています。つまり、最終的には移動するための一般コマンドがかならず必要なのですが、その前に一仕事、いや二仕事でも三仕事でも、好みのことをさせることができるというものです。そして、その内容はアイディア次第で、いくらでも拡張することが可能なのが、このQRLの大きな特徴なのです。このList 6-3において

表 3

サブ・コマンド一覧表

サブ・コマンド	内 容 (得た値は Areg に入る)
DIRME (DIRection to ME)	敵から見て、自機のいる方向番号を得る
SWINGD (SWING Direction)	敵の現在方向を自機の方へ傾け(0 or 1)、その方向番号を得る
SAMDIR (SAmE DIRection)	敵の現在方向番号を得る
WHLR (WHich Left or Right)	自機が敵より右にいれば、キャリーフラクを立てて戻る
WHDU (WHich Down or Up)	自機が敵より下にいれば、キャリーフラクを立てて戻る
SHOTAD (SHOT All Direction)	■方向へ弾を撃つ
LSHOT 1 (Land SHOT 1)	乱数値 < 10H で自機に向けて弾を撃つ(更に発射方向別の制限がある)
LSHOT 2 (Land SHOT 2)	乱数値 < 20H で自機に向けて弾を撃つ(更に発射方向別の制限がある)
LSHOT 3 (Land SHOT 3)	乱数値 < 40H で自機に向けて弾を撃つ(更に発射方向別の制限がある)

は、基本的なものとして、表3のようなサブ・コマンドを用意しましたが、実際に敵のデータがある List 6-4 では、敵の性格に応じていくつか拡張サブ・コマンドを作って、新しい動きをさせています。

前ページのサブ・コマンド一覧表には ESHOT1-3 は入れてありませんが、これをサブ・コマンドとしても使用できるのは当然のことです。また、このサブ・コマンドの内容を見ると、特殊コマンドとの組み合わせが大体想像できると思います。さらに、サブ・コマンドから得たデータによって次に何をさせるのか、そこまで想像できるようになれば、もはや QRL は完璧にマスターしたといえるのですが、ここでそこまで理解できなくても問題はありません。次の List 6-4 は、QRL のオン・パレードになっていますから、その時点で自分のモノにするようにすればいいのです。ただし、サブ・コマンドの QRL プログラムについては、短いものばかりですからキチンと内容を確認しておいてください。

今回のプログラムは、ほとんどが敵の動

きに終始していましたが、6章のスクロール・ゲームにおいては、スクロール・ルーチンとこの QRL が、マスターしたい2大テーマなのです。ここでのテストでは、QRL としては見本にもならないような短いものですが、文法的にはこのような形でプログラム(QRL はアセンブラ上で展開する一種の言語ですから、もはやデータとはいわないのです)を作成していくので、自分の手でもう少し変更してから実験してみてください。なお、テストの実行には List 6-2 でセーブしたプログラムが必要です。そして、このプログラムも次回(List 6-4)のために必要ですので、合わせて再セーブしてください。トータルのセーブ・アドレスは

BD00H-C7D8H

となります。ここでも、マップ・データや敵データは作っていませんので、長くスクロールさせていると、暴走したり、画面が乱れたりすることがあります。動きを確認したら、いつまでも遊んでないで、最後の大仕事にかからなければなりません。

217

List 6-3 敵ロボット言語

アセンブル後、アセンブルした List 6-2 と合わせセーブすること

```

10000          ;***** List 6-3 *****
              ;
BD20          WAIT: EQU 0BD20H
BD2C          SETINT: EQU 0BD2CH
BD43          ORIINT: EQU 0BD43H
BD54          DISP: EQU 0BD54H
BD83          CLPTXY: EQU 0BD83H
BDB0          BLPUT: EQU 0BDB0H
BDFB          CLS: EQU 0BDFBH
BEED          XYADR: EQU 0BEEDH
BF04          PDADR: EQU 0BF04H
BF30          DISPSC: EQU 0BF3DH
BF75          RND: EQU 0BF75H
BF8A          MYMOVE: EQU 0BF8AH

```

List 6-2 で作成したルーチンのアドレス

10140	C0FA	SSKCK: EQU 0C0FAH	
	C138	MYBMOV: EQU 0C138H	
	C16D	SCROLL: EQU 0C16DH	
		:	
		ORG 0C300H	
		:	
	C300	EMAPP: ;EnEMy APPEAR	
	C300 E5	PUSH HL	
	C301 C5	PUSH BC	
	C302 2AD4CE	LD HL, (EDPO)	
	C305 23	INC HL	敵データ・サインを+1する
	C306 22D4CE	LD (EDPO), HL	
	C309 212CCC	LD HL, EMWORK	HL: 敵のワークエリア・出現アドレス
	C30C 111000	LD DE, EMWLEN	DE: 敵のワークエリアの長さ
	C30F 061E	LD B, EMVAL	B: 敵の番号
	C311	EALP: ;Enemy Appear Loop	
	C311 7E	LD A, (HL)	
	C312 B7	OR A	敵のワークエリアの長さかあればEALP
	C313 2806	JR Z, EAOK	
	C315 19	ADD HL, DE	HL: HL + DE 次の敵のワークエリア
	C316 10F9	DJNZ EALP	敵の総数だけEALPを繰り返す
	C318 C1	POP BC	
	C319 E1	POP HL	BC, HLの値をスタックから取り出す
	C31A C9	RET	
		:	
	C31B	EAOK: ;Enemy Appear OK	
	C31B C1	POP BC	BC: 出現初期座標となる
	C31C 36FF	LD (HL), 0FFH	HL: 敵出現中のフラグ
	C31E 23	INC HL	
	C31F ED5BD4CE	LD DE, (EDPO)	DE: 敵データ・サインを+1する
	C323 1A	LD A, (DE)	A: 敵データ・サイン
	C324 77	LD (HL), A	HL: インタで示される敵のパターン番号
	C325 23	INC HL	
	C326 71	LD (HL), C	HL: 敵のワークエリアの初期座標
	C327 23	INC HL	
	C328 70	LD (HL), B	HL: 敵のワークエリアの初期座標
	C329 23	INC HL	
	C32A C5	PUSH BC	
	C32B 87	ADD A, A	A: A * 2 敵パターン番号
	C32C 1100CB	LD DE, EMTTBL	DE: 敵のワークエリアの初期座標(上位)
	C32F 4F	LD C, A	C: A / 2 敵パターン番号
	C330 0600	LD B, 0	B: 敵のワークエリアの初期座標(下位)
	C332 EB	EX DE, HL	DE: 敵のワークエリアの初期座標(上位)
	C333 09	ADD HL, BC	HL: 敵のワークエリアの初期座標(下位)
	C334 EDA0	LDI	
	C336 EDA0	LDI	
	C338 2138CB	LD HL, EMSTBL+2	HL: EMSTBL + 5C 敵を殺した時の座点を示す
	C33B 09	ADD HL, BC	
	C33C EDA0	LDI	
	C33E EDA0	LDI	
	C340 C1	POP BC	BC: 出現初期座標
	C341 C5	PUSH BC	
	C342 0F	RRCA	A: A / 2 敵パターン番号
	C343 CD54BD	CALL DISP	(C, B)にAを表示
	C346 C1	POP BC	
	C347 E1	POP HL	BC, HLの値をスタックから取り出す
	C348 C9	RET	
		:	
	C349	EMMVAL: ;EnEMy MoVe ALI	
	C349 DD212CCC	LD IX, EMWORK	
10740	C34D 061E	LD B, EMVAL	

10750	C34F	EMMLP: ;EneMy Move Loop	
	C34F C5	PUSH BC	
	C350 CD5CC3	CALL EMMOVE	すべての敵をコメントにしたかい移動する
	C353 111000	LD DE,EMWLEN	
	C356 DD19	ADD IX,DE	
	C358 C1	POP BC	
	C359 10F4	DJNZ EMMLP	
	C35B C9	RET	
		; EMMOVE: ;EneMy MOVE	
	C35C		
	C35C DD7E00	LD A,(IX+0)	A ← (IX+0)
	C35F B7	OR A	A=0ならリターン
	C360 C8	RET Z	
	C361 3C	INC A	A=FFHなら ENEMY へ
	C362 2836	JR Z,ENEMY	
	C364 DD4E02	LD C,(IX+2)	敵の X 座標
	C367 DD4603	LD B,(IX+3)	敵の Y 座標
	C36A DD3401	INC (IX+1)	爆発パターンを+1する
	C36D 3C	INC A	
	C36E DD7E01	LD A,(IX+1)	A=FEH なら A ← (IX+1) 後 LANDD へ
	C371 2008	JR NZ,LANDD	
	C373 FE26	CP LDEADP	A<26H なら DISP へ…地上敵の爆発残骸ハタ
	C375 DA54BD	JP C,DISP	…以前、すなわち爆発中のパターン
	C378 C365C5	JP EMFIN	
	C378	LANDD: ;LAND Dead	
	C37B FE27	CP LDEADP+1	A≠27H なら LDPOK へ
	C37D 2003	JR NZ,LDPOK	
	C37F DD3501	DEC (IX+1)	爆発残骸パターンにする
	C382	LDPOK: ;Land Dead Pattern OK	
	C382 3E5C	LD A,DEND	
	C384 B8	CP B	B=5CH(下エンド)なら EMFIN へ
	C385 CA65C5	JP Z,EMFIN	
	C388 C5	PUSH BC	
	C389 210204	LD HL,402H	1コマ移動(下)のための不要部分消去
	C38C CD83BD	CALL CLPTXY	
	C38F C1	POP BC	
	C390 04	INC B	B=B+1…移動後の Y 座標
	C391 DD7003	LD (IX+3),B	
	C394 DD7E01	LD A,(IX+1)	爆発パターン、または残骸パターン
	C397 C354BD	JP DISP	
		; ENEMY: ;ENEMY	
	C39A		
	C39A DD6E04	LD L,(IX+4)	HL ← コマンド・ポインタ
	C39D DD6605	LD H,(IX+5)	
	C3A0	COM: ;COMmand	
	C3A0 4E	LD C,(HL)	C ← コマンド
	C3A1 79	LD A,C	
	C3A2 E680	AND 80H	コマンドのビット7=0なら GENCOM へ
	C3A4 285C	JR Z,GENCOM	
	C3A6 79	LD A,C	
	C3A7 E67F	AND 7FH	A(コマンド)のビット7を0にする
	C3A9 2012	JR NZ,SPCOM	A≠0 なら SPCOM へ
	C3AB DD3600FE	LD (IX+0),0FEH	(IX+0) ← FEH…自爆を意味する
	C3AF 3E20	LD A,EXPP1	A ← 20H…爆発パターン番号
	C3B1 DD7701	LD (IX+1),A	
	C3B4 DD4E02	LD C,(IX+2)	X 座標
	C3B7 DD4603	LD B,(IX+3)	Y 座標
	C3BA C354BD	JP DISP	
		; SPCOM: ;SPeial COMmand	特殊コマンド
	C3BD		
	C3BD 3D	DEC A	A=1 なら JUMPR…コマンドでは 81H
11360	C3BE 283B	JR Z,@JUMPR	

370 C3C0 23	INC HL	
C3C1 5E	LD E,(HL)	DE = コマンドの次の2バイトのデータ
C3C2 23	INC HL	HL = コマンド・ポインタ + 1
C3C3 56	LD D,(HL)	
C3C4 E5	PUSH HL	コマンド・ポインタ + 2 以降
C3C5 EB	EX DE,HL	HL = DE, DE = HL
C3C6 3D	DEC A	A = A - 1
C3C7 2825	JR Z,@IFZR	
C3C9 3D	DEC A	A = A - 1
C3CA 2815	JR Z,@IFCR	A = 3 なら、コマンド・ポインタでは 83H
C3CC 3D	DEC A	A = 4 なら、コマンド・ポインタでは 84H
C3CD 2809	JR Z,@CALLR	
;		
C3CF	@FECHR: ;@FEtCH Routine	A = 5 の時(コマンドでは 85H)
C3CF 01D4C3	LD BC,RET1	HL = 指示レジスタ・ポインタ・レジスタ RET1 から
C3D2 C5	PUSH BC	データをスタックに格納する
C3D3 E9	JP (HL)	ジャンプ先は HL
C3D4	RET1: ;RETurn 1	
C3D4 E1	POP HL	
C3D5 4F	LD C,A	
C3D6 182A	JR GENCOM	
;		
C3D8	@CALLR: ;@CALL Routine	
C3D8 01D0C3	LD BC,RET2	
C3DB C5	PUSH BC	
C3DC E9	JP (HL)	
C3DD	RET2: ;RETurn 2	
C3DD E1	POP HL	スタックからデータをとり出す
C3DE 23	INC HL	
C3DF 18BF	JR COM	
;		
C3E1	@IFCR: ;@IFC Routine	
C3E1 01E6C3	LD BC,RET3	
C3E4 C5	PUSH BC	BC = CALI
C3E5 E9	JP (HL)	
C3E6	RET3: ;RETurn 3	
C3E6 E1	POP HL	コマンド・ポインタを取り出す
C3E7 3812	JR C,@JUMPR	もし、条件が満たされなければ JUMPR
C3E9 23	INC HL	
C3EA 23	INC HL	
C3EB 23	INC HL	HL = HL + 3 IMF 用データ・レジスタ
C3EC 18B2	JR COM	次のコマンド・ポインタにする
;		
C3EE	@IFZR: ;@IFZ Routine	
C3EE 01F3C3	LD BC,RET4	
C3F1 C5	PUSH BC	BC = CALI HL
C3F2 E9	JP (HL)	
C3F3	RET4: ;RETurn 4	
C3F3 E1	POP HL	コマンド・ポインタを取り出す
C3F4 2805	JR Z,@JUMPR	ゼロフラグが立っていれば JUMPR へ
C3F6 23	INC HL	
C3F7 23	INC HL	HL = HL + 2 UMF 用のデータをスキップ
C3F8 23	INC HL	し、次のコマンド・ポインタにする
C3F9 18A5	JR COM	
;		
C3FB	@JUMPR: ;@JUMP Routine	
C3FB 23	INC HL	
C3FC 5E	LD E,(HL)	HL = コマンドの次の2バイトのデータ
C3FD 23	INC HL	新コマンド・ポインタとなる
C3FE 56	LD D,(HL)	
C3FF EB	EX DE,HL	
11980 C400 189E	JR COM	

C402		; GENCOM: ; GENeral COMmand	一般コメント
C402 23	INC	HL	
C403 DD7504	LD	(IX+4),L	3回のため
C406 DD7405	LD	(IX+5),H	
C409 79	LD	A,C	
C40A E630	AND	30H	
C40C 2811	JR	Z,ONLYM	
C40E 211FC4	LD	HL,ONLYM	
C411 E5	PUSH	HL	
C412 FE10	CP	10H	
C414 CA6EC6	JP	Z,ESHOT1	
C417 FE20	CP	20H	
C419 CA76C6	JP	Z,ESHOT2	
C41C C37CC6	JP	ESHOT3	
C41F	; ONLYM: ; ONLY Move		
C41F 79	LD	A,C	
C420 DD4E02	LD	C,(IX+2)	
C423 DD4603	LD	B,(IX+3)	
C426 E60F	AND	0FH	
C428 DD7708	LD	(IX+8),A	
C42B 6F	LD	L,A	
C42C 87	ADD	A,A	
C42D 85	ADD	A,L	
C42E 2136C4	LD	HL,EMTBL	
C431 5F	LD	E,A	
C432 1600	LD	D,0	
C434 19	ADD	HL,DE	
C435 E9	JP	(HL)	
C436	; EMTBL: ; EneMy TaBLE		
C436 C351C4	JP	EMSTOP	
C439 C363C4	JP	EMRR	
C43C C384C4	JP	EMUR	
C43F C3A8C4	JP	EMUU	
C442 C3BDC4	JP	EMUL	
C445 C3E1C4	JP	EMLL	
C448 C303C5	JP	EMDL	
C44B C326C5	JP	EMDD	
C44E C337C5	JP	EMDR	
C451	; EMSTOP: ; EneMy STOP		
C451 3E5C	LD	A,DEND	
C453 B8	CP	B	
C454 CA65C5	JP	Z,EMFIN	
C457 C5	PUSH	BC	
C458 210204	LD	HL,402H	
C45B CD83BD	CALL	CLPTXY	
C45E C1	POP	BC	
C45F 04	INC	B	
C460 C354C5	JP	EMDISP	
C463	; EMRR: ; EneMy direction=RR		
C463 3E30	LD	A,REND	
C465 B9	CP	C	
C466 CA65C5	JP	Z,EMFIN	
C469 3E5C	LD	A,DEND	
C46B B8	CP	B	
C46C CA65C5	JP	Z,EMFIN	
C46F C5	PUSH	BC	
12600 C470 210204	LD	HL,402H	

12610	C473 CD83BD	CALL CLPTXY	
	C476 C1	POP BC	移動方向=1の場合の不要部分消去+次座標計算
	C477 04	INC B	
	C478 C5	PUSH BC	
	C479 210E01	LD HL,10EH	
	C47C CD83BD	CALL CLPTXY	
	C47F C1	POP BC	
	C480 0C	INC C	
	C481 C354C5	JP EMDISP	
	C484	;EMUR: ;EnemY direction=UR	
	C484 3E30	LD A,REND	
	C486 B9	CP C	C=30H(右エッジ)なら EMFIN
	C487 CA65C5	JP Z,EMFIN	
	C48A AF	XOR A	
	C48B B8	CP B	B=0(上エッジ)なら EMFIN
	C48C CA65C5	JP Z,EMFIN	
	C48F C5	PUSH BC	移動方向=2の場合の不要部分消去+次座標計算
	C490 210E01	LD HL,10EH	
	C493 CD83BD	CALL CLPTXY	
	C496 C1	POP BC	
	C497 C5	PUSH BC	
	C498 3E07	LD A,7	
	C49A 80	ADD A,B	
	C49B 47	LD B,A	
	C49C 210204	LD HL,402H	
	C49F CD83BD	CALL CLPTXY	
	C4A2 C1	POP BC	
	C4A3 0C	INC C	
	C4A4 05	DEC B	
	C4A5 C354C5	JP EMDISP	
	C4A8	;EMUU: ;EnemY direction=UU	
	C4A8 AF	XOR A	
	C4A9 B8	CP B	
	C4AA CA65C5	JP Z,EMFIN	
	C4AD C5	PUSH BC	移動方向=3の場合の不要部分消去+次座標計算
	C4AE 3E07	LD A,7	
	C4B0	ADD A,B	
	C4B1 47	LD B,A	
	C4B2 210204	LD HL,402H	
	C4B5 CD83BD	CALL CLPTXY	
	C4B8 C1	POP BC	
	C4B9 05	DEC B	
	C4BA C354C5	JP EMDISP	
	C4BD	;EMUL: ;EnemY direction=UL	
	C4BD AF	XOR A	
	C4BE B9	CP C	
	C4BF CA65C5	JP Z,EMFIN	
	C4C2 B8	CP B	
	C4C3 CA65C5	JP Z,EMFIN	
	C4C6 C5	PUSH BC	
	C4C7 0C	INC C	
	C4C8 0C	INC C	
	C4C9 0C	INC C	
	C4CA 210E01	LD HL,10EH	
	C4CD CD83BD	CALL CLPTXY	
	C4D0 C1	POP BC	
	C4D1 C5	PUSH BC	
	C4D2 3E07	LD A,7	
	C4D4 80	ADD A,B	
13230	C4D5 47	LD B,A	

124	C4D6 210204	LD HL,402H	
	C4D9 CD83BD	CALL CLPTXY	
	C4DC C1	POP BC	
	C4DD 0D	DEC C	
	C4DE 05	DEC B	
	C4DF 1873	JR EMDISP	
	; EMLL: ;Enemy direction=LL		
	C4E1	XOR A	
	C4E1 AF	CP C	左エ(1)なら EMFINへ
	C4E2 B9	JP Z,EMFIN	
	C4E3 CA65C5	LD A,DEND	
	C4E6 3E5C	CP B	下エ(1)なら EMFINへ
	C4E8 B8	JP Z,EMFIN	
	C4E9 CA65C5	PUSH BC	移動方向=5の場合の不要部分消去+次座標計算
	C4EC C5	INC C	
	C4ED 0C	INC C	
	C4EE 0C	INC C	
	C4EF 0C	LD HL,110H	
	C4F0 211001	CALL CLPTXY	
	C4F3 CD83BD	POP BC	
	C4F6 C1	PUSH BC	
	C4F7 C5	LD HL,302H	
	C4F8 210203	CALL CLPTXY	
	C4FB CD83BD	POP BC	
	C4FE C1	INC B	
	C4FF 04	DEC C	C,B) → (C-1,B+1)
	C500 0D	JR EMDISP	
	C501 1851		
	; EMDL: ;Enemy direction=DL		
	C503	LD A,DEND-2	
	C503 3E5A	CP B	上エ(1)なら EMFINへ
	C505 B8	JR C,EMFIN	
	C506 385D	XOR A	
	C508 AF	CP C	左エ(1)なら EMFINへ
	C509 B9	JP Z,EMFIN	
	C50A 2859	JR Z,EMFIN	
	C50C C5	PUSH BC	移動方向=6の場合の不要部分消去+次座標計算
	C50D 210404	LD HL,404H	
	C510 CD83BD	CALL CLPTXY	
	C513 C1	POP BC	
	C514 C5	PUSH BC	
	C515 0C	INC C	
	C516 0C	INC C	
	C517 0C	INC C	
	C518 04	INC B	
	C519 04	INC B	
	C51A 210C01	LD HL,10CH	
	C51D CD83BD	CALL CLPTXY	
	C520 C1	POP BC	
	C521 0D	DEC C	
	C522 04	INC B	
	C523 04	INC B	
	C524 182E	JR EMDISP	
	; EMDD: ;Enemy direction=DD		
	C526	LD A,DEND-2	
	C526 3E5A	CP B	上エ(1)なら EMFINへ
	C528 B8	JR C,EMFIN	
	C529 383A	PUSH BC	移動方向=7の場合の不要部分消去+次座標計算
	C52B C5	LD HL,404H	
	C52C 210404	CALL CLPTXY	
	C52F CD83BD	POP BC	
	C532 C1	INC B	
13860	C533 04		

C534 04	INC B		
C535 181D	JR EMDISP		
;			
C537	EMDR: ;EneMy direction=DR		
C537 3E30	LD A,REND		
C539 B9	CP C		
C53A 2829	JR Z,EMFIN		
C53C 3E5A	LD A,DEND-2		
C53E B8	CP B		
C53F 3824	JR C,EMFIN		
C541 C5	PUSH BC		
C542 211001	LD HL,110H		
C545 CD83BD	CALL CLPTXY		
C548 C1	POP BC		
C549 0C	INC C		
C54A C5	PUSH BC		
C54B 210403	LD HL,304H		
C54E CD83BD	CALL CLPTXY		
C551 C1	POP BC		
C552 04	INC B		
C553 04	INC B		
C554	EMDISP: ;EneMy DISPlay		
C554 DD7102	LD (IX+2),C		
C557 DD7003	LD (IX+3),B		
C55A C5	PUSH BC		
C55B CD6FC5	CALL MYBCHK		
C55E C1	POP BC		
C55F DD7E01	LD A,(IX+1)		
C562 C354BD	JP DISP		
;			
C565	EMFIN: ;EneMy FINish		
C565 DD360000	LD (IX+0),0		
C569 211004	LD HL,410H		
C56C C383BD	JP CLPTXY		
;			
C56F	MYBCHK: ;MY Bullet Check with enemy		
C56F 060C	LD B,MYBVAL		
C571 21ACCE	LD HL,MYBWK		
C574 110300	LD DE,MYBWL		
C577	MBCLP: ;MyBchk Loop		
C577 7E	LD A,(HL)		
C578 B7	OR A		
C579 283E	JR Z,NTMB2		
C57B E5	PUSH HL		
C57C 23	INC HL		
C57D 7E	LD A,(HL)		
C57E DD9602	SUB (IX+2)		
C581 FE04	CP 4		
C583 3033	JR NC,NTMB1		
C585 23	INC HL		
C586 7E	LD A,(HL)		
C587 DD9603	SUB (IX+3)		
C58A FE07	CP 7		
C58C 302A	JR NC,NTMB1		
C58E DD7E01	LD A,(IX+1)		
C591 FE04	CP 4		
C593 3EFE	LD A,0FEH		
C595 3001	JR NC,EXPSET		
C597 3D	DEC A		
C598	EXPSET: ;EXPlosion SET		
C598 DD7700	LD (IX+0),A		
C598 DD360120	LD (IX+1),EXPP1		

C59F 3AC1E6	LD A,(0E6C1H)	
C5A2 F620	OR 20H	
C5A4 D340	OUT (40H),A	
C5A6 DD5E06	LD E,(IX+6)	エのX座標に6を加算
C5A9 DD5607	LD D,(IX+7)	
C5AC CD3DBF	CALL DISPSC	
C5AF 3AC1E6	LD A,(0E6C1H)	REP
C5B2 D340	OUT (40H),A	
C5B4 E1	POP HL	
C5B5 3600	LD (HL),0	敵の弾の移動方向を0にする
C5B7 C9	RET	
C5B8	NTMB1: ;Next My Bullet 1	
C5B8 E1	POP HL	
C5B9	NTMB2: ;Next My Bullet 2	
C5B9 19	ADD HL,DE	エの自機が弾のワークエリア
C5BA 10BB	DJNZ MBCLP	
C5BC C9	RET	
C5BD	; EMBMOV: ;Enemy Bullet MOVE	敵の弾の移動
C5BD 210CCE	LD HL,EMBWK	エの弾のワークエリアをHLに格納
C5C0 0628	LD B,EMBVAL	エの弾の移動方向をBに格納
C5C2	EBMLP: ;EmbMov Loop	
C5C2 7E	LD A,(HL)	
C5C3 B7	OR A	敵の弾が出現中ならEBINGをコールする
C5C4 C4CEC5	CALL NZ,EBING	
C5C7 110400	LD DE,EMBULE	HL・HL+エの弾のワークエリア
C5CA 19	ADD HL,DE	
C5CB 10F5	DJNZ EBMLP	
C5CD C9	RET	
C5CE	; EBING: ;Enemy Bullet is moving	
C5CE C5	PUSH BC	
C5CF 23	INC HL	
C5D0 4E	LD C,(HL)	エの弾の座標
C5D1 23	INC HL	エの弾の座標
C5D2 46	LD B,(HL)	
C5D3 23	INC HL	
C5D4 E5	PUSH HL	
C5D5 C5	PUSH BC	
C5D6 210401	LD HL,104H	エにある敵の弾を消去
C5D9 CD83BD	CALL CLPTXY	
C5DC C1	POP BC	
C5DD E1	POP HL	
C5DE 7E	LD A,(HL)	
C5DF 87	ADD A,A	HL・敵の弾の移動方向×2
C5E0 E5	PUSH HL	
C5E1 21E9C5	LD HL,EMBTBL	
C5E4 5F	LD E,A	HL・EMBTBL+A→移動方向別のジャンプ・テーブル・アドレスを示す
C5E5 1600	LD D,0	
C5E7 19	ADD HL,DE	
C5E8 E9	JP (HL)	
C5E9	EMBTBL: ;Enemy Bullet Table	
C5E9 1858	JR EMBDD	
C5EB 180E	JR EMBRR	
C5ED 181A	JR EMBUR	敵の弾の移動方向別のジャンプ・テーブル
C5EF 1825	JR EMBUU	*エの方向あるのは、敵が停止中に弾を発射した時に、移動方向=0となるため、そこで、移動方向=エと同じにする
C5F1 182A	JR EMBUL	
C5F3 1833	JR EMBLL	
C5F5 183E	JR EMBDL	
C5F7 184A	JR EMBDD	
TS100 C5F9 1851	JR EMBDR	

15110

```

;
;EMBRR: ;EnEMy Bullet direction=RR
C5FB      LD    A,REND+3
C5FB 3E33  CP    C
C5FD B9    JR    Z,EMBFIN      C=33H(右エント+3)なら EMBFINへ
C5FE 2866  LD    A,DEND+6
C600 3E62  CP    B
C602 B8    JR    Z,EMBFIN      B=62H(下エント+6)なら EMBFINへ
C603 2861  INC    C
C605 0C    ;
C606 04    INC    B
C607 1850  JR    EBDISP      (C,B) → (C+1,B+1)

```

```

;
;EMBUR: ;EnEMy Bullet direction=UR
C609      XOR    A
C609 AF    CP    B
C60A B8    JR    Z,EMBFIN      B=0(上エント)なら EMBFINへ
C60B 2859  LD    A,REND+3
C60D 3E33  CP    C
C60F B9    JR    Z,EMBFIN      C=33H(右エント+3)なら EMBFINへ
C610 2854  INC    C
C612 0C    ;
C613 05    DEC    B
C614 1843  JR    EBDISP      (C,B) → (C+1,B-1)

```

```

;
;EMBUU: ;EnEMy Bullet direction=UU
C616      XOR    A
C616 AF    CP    B
C617 B8    JR    Z,EMBFIN      B=0(上エント)なら EMBFINへ
C618 284C  DEC    B
C61A 05    JR    EBDISP      (C,B) → (C,B-1)
C61B 183C

```

```

;
;EMBUL: ;EnEMy Bullet direction=UL
C61D      XOR    A
C61D AF    CP    B
C61E B8    JR    Z,EMBFIN      B=0(上エント)なら EMBFINへ
C61F 2845  CP    C
C621 B9    JR    Z,EMBFIN      C=0(左エント)なら EMBFINへ
C622 2842  DEC    C
C624 0D    DEC    B
C625 05    JR    EBDISP      (C,B) → (C-1,B-1)
C626 1831

```

```

;
;EMBL: ;EnEMy Bullet direction=LL
C628      XOR    A
C628 AF    CP    C
C629 B9    JR    Z,EMBFIN      C=0(左エント)なら EMBFINへ
C62A 283A  LD    A,DEND+6
C62C 3E62  CP    B
C62E B8    JR    Z,EMBFIN      B=62H(下エント+6)なら EMBFINへ
C62F 2835  DEC    C
C631 0D    DEC    B
C632 04    JR    EBDISP      (C,B) → (C-1,B+1)
C633 1824

```

```

;
;EMBDL: ;EnEMy Bullet direction=DL
C635      XOR    A
C635 AF    CP    C
C636 B9    JR    Z,EMBFIN      C=0(左エント)なら EMBFINへ
C637 282D  LD    A,DEND+5
C639 3E61  CP    B
C63B B8    JR    C,EMBFIN      B>61H(下エント+5)なら EMBFINへ
C63C 3828  DEC    C
C63E 0D    ;
C63F 04    INC    B
C640 04    INC    B
C641 1816  JR    EBDISP      B1 → (C-1,B+2)

```

15720

15730

```

;
EMBDD: ;EnemY Bullet direction=DD
C643 LD A,DEND+5
C643 3E61 CP B
C645 B8 JR C,EMBFIN
C646 381E INC B
C648 04 INC B
C649 04 JR EBDISP
C64A 180D

;
EMBDR: ;EnemY Bullet direction=DR
C64C LD A,DEND+5
C64C 3E61 CP B
C64E B8 JR C,EMBFIN
C64F 3815 LD A,REND+3
C651 3E33 CP C
C653 B9 JR Z,EMBFIN
C654 2810 INC C
C656 0C
C657 04 INC B
C658 04 INC B
C659 EBDISP: ;EnemY Bullet DISPlay
C659 E1 POP HL
C65A 2B DEC HL
C65B 70 LD (HL),B
C65C 2B DEC HL
C65D 71 LD (HL),C
C65E E5 PUSH HL
C65F CDB0BD CALL BLPUT
C662 E1 POP HL
C663 2B DEC HL
C664 C1 POP BC
C665 C9 RET

;
EMBFIN: ;EnemY Bullet FINish
C666 POP HL
C666 E1 DEC HL
C667 2B DEC HL
C668 2B DEC HL
C669 2B DEC HL
C66A 3600 LD (HL),0
C66C C1 POP BC
C66D C9 RET

;
ESHOT1: ;Enemy SHOT 1
C66E CALL RND
C66E CD75BF CP 5
C671 FE05 JR C,ESHOT3
C673 3807 RET
C675 C9
C676 ESHOT2: ;Enemy SHOT 2
C676 CD75BF CALL RND
C679 FE20 CP 20H
C67B D0 RET NC
C67C ESHOT3: ;Enemy SHOT 3
C67C 0628 LD B,EMBVAl
C67E 210CCE LD HL,EMBWOK
C681 110400 LD DE,EMBWLE
C684 ESLP: ;Enemy Shot Loop
C684 7E LD A,(HL)
C685 B7 OR A
C686 2804 JR Z,ESOK
C688 19 ADD HL,DE
C689 10F9 DJNZ ESLP
C68B C9 RET

```

16330

B>61H(下エンド+5)なら EMBFIN へ

(C,B) → (C,B+2) へ

B>61H(下エンド+5)なら EMBFIN へ

C=33H(右エンド+3)なら EMBFIN へ

(C,B) → (C+1,B+2)

HL の値をスタックから取り出す

移動後の弾の座標をワークエリアにストア

(C,B)に弾を表示
HL はワークエリアの先頭になっている

弾の出現フラグを 0 にする

乱数値<5 なら ESHOT3 へ
…弾発射の確率=5/256

乱数値≥20H ならリターン
…弾発射の確率=32/256

B・ 弾の総数
HL・ 弾のワークエリア・先頭アドレス
DE・ 弾1発のワークエリアの長さ

敵の弾のワークエリアに空きがあればESOKへ

次の敵の弾のワークエリアにする
敵の弾の総数だけESLPを繰り返す

16340	C68C	ESOK:	LD	A,(IX+8)	敵の移動方向
	C68C DD7E08		CP	5	
	C68F FE05		JR	C,ED1234	
	C691 380E		CP	7	
	C693 FE07		JR	C,ED56	
	C695 3805				
	C697 110206		LD	DE,602H	弾出現座標のオフセット値
	C69A 1811		JR	DTSET	
	C69C	ED56: ;Enemy Direction=5,6			
	C69C 110004		LD	DE,400H	弾出現座標のオフセット値
	C69F 180C		JR	DTSET	
	C6A1	ED1234: ;Enemy Direction=1,2,3,4			
	C6A1 FE03		CP	3	
	C6A3 3805		JR	C,ED12	
	C6A5 110100		LD	DE,001H	弾出現座標のオフセット値
	C6A8 1803		JR	DTSET	
	C6AA	ED12: ;Enemy Direction=1,2			
	C6AA 110303		LD	DE,303H	弾出現座標のオフセット値
		DTSET: ;Data SET			
	C6AD		LD	B,A	
	C6AD 47		LD	(HL),1	
	C6AE 3601		INC	HL	
	C6B0 23		LD	A,(IX+2)	敵のX座標
	C6B1 DD7E02		ADD	A,E	
	C6B4 83		LD	(HL),A	
	C6B5 77		INC	HL	
	C6B6 23		LD	A,(IX+3)	(HL+4)初期Y座標
	C6B7 DD7E03		ADD	A,D	
	C6BA 82		LD	(HL),A	敵のY座標
	C6BB 77		INC	HL	
	C6BC 23		LD	(HL),B	(HL+4)移動方向
	C6BD 70		RET		
	C6BE C9				
		DIRME: ;DIRECTION to ME			
	C6BF		LD	HL,(MYLOC)	—自機のいる方向番号を得る
	C6BF 2AD0CE		LD	A,(IX+2)	敵のX座標
	C6C2 DD7E02		SUB	L	敵のY座標
	C6C5 95		JR	NC,ELEFT	A≥0ならELEFT
	C6C6 3027				
	C6C8	ERIGHT: ;Enemy is RIGHT			
	C6C8 ED44		NEG		
	C6CA 4F		LD	C,A	敵のX座標
	C6CB DD7E03		LD	A,(IX+3)	敵のY座標
	C6CE 94		SUB	H	A≥0ならELEFT
	C6CF 3016		JR	NC,ERND	
	C6D1	ERND: ;Enemy is Right and Down			
	C6D1 ED44		NEG		
	C6D3 47		LD	B,A	
	C6D4 79		LD	A,C	
	C6D5 87		ADD	A,A	
	C6D6 87		ADD	A,A	
	C6D7 88		CP	B	
	C6D8 3E07		LD	A,7	
	C6DA 08		RET	C	
	C6DB 79		LD	A,C	
	C6DC 87		ADD	A,A	
	C6DD 81		ADD	A,C	
	C6DE CB20		SLA	B	
	C6E0 88		CP	B	
	C6E1 3E08		LD	A,8	
	C6E3 08		RET	C	
	C6E4 3E01		LD	A,1	
16960	C6E6 C9		RET		

敵の移動方向

D123

弾出現座標のオフセット値

弾出現座標のオフセット値

敵の移動方向

敵の移動方向

弾出現座標のオフセット値

弾出現座標のオフセット値

p 234

敵の移動方向

敵の移動方向

敵のX座標

敵のX座標

(HL+4)初期Y座標

敵のY座標

(HL+4)移動方向

—自機のいる方向番号を得る

敵のX座標

敵のY座標

敵のY座標

A≥0ならELEFT

敵のX座標

敵のY座標

敵のY座標

A≥0ならELEFT

敵のX座標

敵のY座標

敵のX座標

敵のY座標

敵のY座標

敵のY座標

敵のY座標

敵のY座標

敵のY座標

敵のY座標

敵のY座標

```

C6E7          ERNU: ;Enemy's Right aNd Up
C6E7 CB21     SLA  C
C6E9 B9       CP   C
C6EA 3E02     LD   A,2
C6EC D8       RET  C
C6ED 3C       INC  A
C6EE C9       RET

```

```

C6EF          ;ELEFT: ;Enemy's LEFT
C6EF 4F       LD   C,A
C6F0 DD7E03   LD   A,(IX+3)
C6F3 94       SUB  H
C6F4 3015     JR   NC,ELNU

```

```

C6F6          ELND: ;Enemy's Left aNd Down
C6F6 ED44     NEG
C6F8 47       LD   B,A
C6F9 79       LD   A,C
C6FA 87       ADD  A,A
C6FB 87       ADD  A,A
C6FC 88       CP   B
C6FD 3E07     LD   A,7
C6FF D8       RET  C
C700 79       LD   A,C
C701 87       ADD  A,A
C702 81       ADD  A,C
C703 CB20     SLA  B
C705 B8       CP   B
C706 3E06     LD   A,6
C708 D8       RET  C
C709 3D       DEC  A
C70A C9       RET

```

```

C70B          ELNU: ;Enemy's Left aNd Up
C70B CB21     SLA  C
C70D B9       CP   C
C70E 3E04     LD   A,4
C710 D8       RET  C
C711 3D       DEC  A
C712 C9       RET

```

```

C713          ;SWINGD: ;SWING Direction
C713 CDBFC6   CALL DIRME
C716 DD4608   LD   B,(IX+8)
C719 90       SUB  B
C71A 281A     JR   Z,NOSWG
C71C FA25C7   JP   M,SWGM
C71F FE04     CP   4
C721 300B     JR   NC,DECSWG
C723 1804     JR   INCSWG

```

```

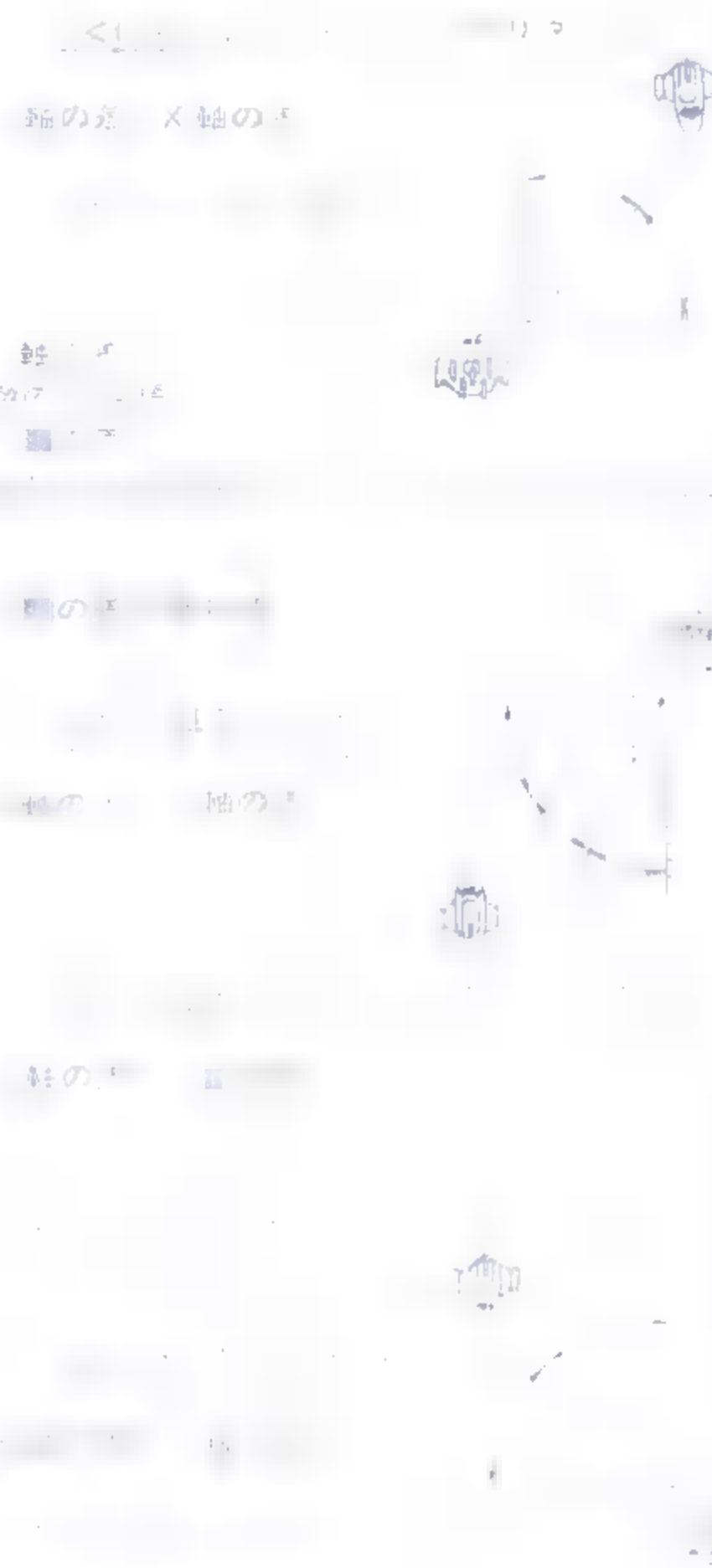
C725          SWGM: ;SWing Minus
C725 FEFC     CP   -4
C727 3005     JR   NC,DECSWG
C729          INCSWG: ;INCrement SWing
C729 78       LD   A,B
C72A E607     AND  7
C72C 3C       INC  A
C72D C9       RET

```

```

C72E          DECSWG: ;DECrement SWing
C72E 78       LD   A,B
C72F 3D       DEC  A
C730 E607     AND  7
C732 C0       RET  NZ
C733 3E08     LD   A,8
C735 C9       RET

```



17600	C736	NOSWG: ;NO SWinG	
	C736 78	LD A,B	方向の変更なし
	C737 C9	RET	
	C738	;SAMDIR: ;SAME DIRection	——敵の方向番号を得る
	C738 DD7E08	LD A,(IX+8)	■の移動方向
	C73B C9	RET	
	C73C	;WHLR: ;WHich Left or Right	
	C73C 3AD0CE	LD A,(MYLOC)	
	C73F DD8E02	CP (IX+2)	自機が敵より右にいればキャリーフラグが立つ
	C742 3F	CCF	
	C743 C9	RET	
	C744	;WHDU: ;WHich Down or Up	
	C744 3AD1CE	LD A,(MYLOC+1)	
	C747 DD8E03	CP (IX+3)	自機が敵より下にいればキャリーフラグが立つ
	C74A C9	RET	
	C74B	;SHOTAD: ;SHOT All Direction	
	C74B 0E08	LD C,8	C・■の弾発射の数、および方向を示す
	C74D	SADLP: ;ShotAD Loop	
	C74D 79	LD A,C	
	C74E 328EC7	LD (BDFIX+1),A	(BDFIX+1)・A=弾の発射予定方向設定
	C751 C5	PUSH BC	敵の弾のワークエリアに空きがあれば、発射の
	C752 CD8DC7	CALL BDFIX	設定をし、なければキャリーフラグを立てて戻
	C755 C1	POP BC	てくる
	C756 D8	RET C	キャリーフラグが立っていればリターン
	C757 0D	DEC C	
	C758 20F3	JR NZ,SADLP	SADLPを8回繰り返す
	C75A C9	RET	
	C75B	;LSHOT1: ;Land SHOT 1	
	C75B CD75BF	CALL RND	
	C75E FE10	CP 10H	乱数値≥10Hならリターン
	C760 D0	RET NC	
	C761 180E	JR PBYD	
	C763	;LSHOT2: ;Land SHOT 2	
	C763 CD75BF	CALL RND	
	C766 FE20	CP 20H	乱数値≥20Hならリターン
	C768 D0	RET NC	
	C769 1806	JR PBYD	
	C76B	;LSHOT3: ;Land SHOT 3	
	C76B CD75BF	CALL RND	
	C76E FE40	CP 40H	乱数値≥40Hならリターン
	C770 D0	RET NC	
	C771	;PBYD: ;Possibility BY Direction	
	C771 CDBFC6	CALL DIRME	・目標の方向番号と弾の方向番号を比較する
	C774 328EC7	LD (BDFIX+1),A	弾の発射予定方向番号
	C777 E607	AND 7	
	C779 D603	SUB 3	A=1 2 3 4 5 6 7 8カ
	C77B 3002	JR NC,CKPOS	
	C77D ED44	NEG	0 1 2 3 4 5 6 7 8
	C77F	CKPOS: ;Check POSSibility	・234 参照
	C77F 3C	INC A	
	C780 6F	LD L,A	
	C781 CD75BF	CALL RND	・乱数値を求め
	C784 E603	AND 3	
	C786 BD	CP L	
18220	C787 D0	RET NC	

18230	C788 CD75BF	CALL RND	
	C78B 17	RLA	乱数値≥80Hならリターン…フィルタ②
	C78C 08	RET C	
	C78D	;BDFIX: ;Bullet Direction FIX	
	C78D 3E00	LD A,0	A・弾の発射予定方向…0はダミー
	C78F FE05	CP 5	A<5ならBD1234へ
	C791 380E	JR C,BD1234	
	C793 FE07	CP 7	A<7ならBD56へ
	C795 3805	JR C,BD56	
	C797 210206	LD HL,602H	HL・0602H…弾出現座標のオフセット値
	C79A 1811	JR BPSET	
	C79C	BD56: ;Bullet Direction=5,6	
	C79C 210004	LD HL,400H	HL・0400H…弾出現座標のオフセット値
	C79F 180C	JR BPSET	
	C7A1	BD1234: ;Bullet Direction=1,2,3,4	
	C7A1 FE03	CP 3	A<3ならBD12へ
	C7A3 3805	JR C,BD12	
	C7A5 210100	LD HL,001H	HL・0001H…弾出現座標のオフセット値
	C7A8 1803	JR BPSET	
	C7AA	BD12: ;Bullet Direction=1,2	
	C7AA 210303	LD HL,303H	HL・0303H…弾出現座標のオフセット値
	C7AD	BPSET: ;Bullet Position SET	
	C7AD 22C8C7	LD (SETPD+1),HL	
	C7B0 4F	LD C,A	C・A…弾の発射予定方向
	C7B1	;SKSWK: ;Seek Shot Work area	
	C7B1 0628	LD B,EMBVAL	B・敵の弾の総数
	C7B3 210CCE	LD HL,EMBWK	HL・敵の弾のワークエリア先頭アドレス
	C7B6 110400	LD DE,EMBWL	DE・敵の弾1発のワークエリアの長さ
	C7B9	SKSLP: ;SKSwk Loop	
	C7B9 7E	LD A,(HL)	
	C7BA B7	OR A	
	C7BB 2805	JR Z,SHOTOK	敵の弾のワークエリアに空きがあればSHOTOKへ
	C7BD 19	ADD HL,DE	なければ、キャリーフラグを立ててリターン
	C7BE 10F9	DJNZ SKSLP	
	C7C0 37	SCF	
	C7C1 C9	RET	
	C7C2	SHOTOK: ;SHOT OK	
	C7C2 C5	PUSH BC	は弾の発射予定方向
	C7C3 3601	LD (HL),1	敵の弾出現フラグのセット…1
	C7C5 23	INC HL	
	C7C6 EB	EX DE,HL	
	C7C7	SETPD: ;SET Position & Direction	
	C7C7 010000	LD BC,0000	BC・弾出現座標のオフセット値…0000はダミー
	C7CA DD6E02	LD L,(IX+2)	敵のX座標
	C7CD DD6603	LD H,(IX+3)	敵のY座標
	C7D0 09	ADD HL,BC	弾発射の初期座標
	C7D1 EB	EX DE,HL	
	C7D2 73	LD (HL),E	弾の初期X座標
	C7D3 23	INC HL	
	C7D4 72	LD (HL),D	弾の初期Y座標
	C7D5 23	INC HL	
	C7D6 C1	POP BC	
	C7D7 71	LD (HL),C	弾の発射方向
	C7D8 C9	RET	
		;ORG 0C800H	
	C800	;TEST: ;TEST	
	C800 F3	OI	
	C801 AF	XOR A	初期設定
18850	C802 0351	OUT (51H),A	

```

C804 3100B6      LD  SP,0B600H
C807 CDFBBD      CALL CLS

;
C80A 2AD9CE      LD  HL,(SCENT)
C80D 0600        LD  B,0
C80F             MAPLP1: ;MAPPING Loop 1
C80F 3604        LD  (HL),4
C811 23          INC  HL
C812 10FB        DJNZ MAPLP1

;
C814 2AD9CE      LD  HL,(SCENT)
C817 0600        LD  B,0
C819             MAPLP2: ;MAPPING Loop 2
C819 2B          DEC  HL
C81A 3604        LD  (HL),4
C81C 2B          DEC  HL
C81D 3684        LD  (HL),84H
C81F 2B          DEC  HL
C820 3603        LD  (HL),3
C822 2B          DEC  HL
C823 3603        LD  (HL),3
C825 2B          DEC  HL
C826 3603        LD  (HL),3
C828 10EF        DJNZ MAPLP2

;
C82A 2AD9CE      LD  HL,(SCENT)
C82D 22D6CE      LD  (SCTOP),HL
C830 AF          XOR  A
C831 32D8CE      LD  (SCCT),A
C834 2ADBCE      LD  HL,(EMENT)
C837 2B          DEC  HL
C838 22D4CE      LD  (EDPO),HL

;
C83B 21165B      LD  HL,5B16H
C83E 22D0CE      LD  (MYLOC),HL
C841 21ACCE      LD  HL,MYBWOK
C844 110300      LD  DE,MYBWLE
C847 060C        LD  B,MYBVAL
C849             IMYBLP: ;Initialize My Bullet Loop
C849 3600        LD  (HL),0
C84B 19          ADD  HL,DE
C84C 10FB        DJNZ IMYBLP

;
C84E 210CCE      LD  HL,EMBWOK
C851 110400      LD  DE,EMBWLE
C854 0628        LD  B,EMBVAL
C856             IEMBLP: ;Initialize Enemy Bullet Loop
C856 3600        LD  (HL),0
C858 19          ADD  HL,DE
C859 10FB        DJNZ IEMBLP

;
C85B 212CCC      LD  HL,EMWORK
C85E 111000      LD  DE,EMWLEN
C861 061E        LD  B,EMVAL
C863             IEMLP: ;Initialize Enemy Loop
C863 3600        LD  (HL),0
C865 19          ADD  HL,DE
C866 10FB        DJNZ IEMLP
C868 CD2CBD      CALL SETINT

;
C86B             MAIN: ;MAIN loop
C86B CDFAC0      CALL SSKCK
C86E CD38C1      CALL MYBMOV

```



```

C871 CDBDC5      CALL EMBMOV
C874 CD6DC1      CALL SCROLL
C877 CD49C3      CALL EMMVAL
C87A CD8ABF      CALL MYMOVE
C87D CD38C1      CALL MYBMOV
C880 CDBDC5      CALL EMBMOV
C883 CD20BD      CALL WAIT

;
M1: ;Main 1
C886             IN      A,(8)
C886 DB08        RRA
C888 1F          RRA
C889 38E0        JR      C,MAIN
C88B 1F          RRA
C88C 38F8        JR      C,M1
C88E CD43BD      CALL ORIINT
C891 FF          RST 38H

;
ORG 0CB00H

;
EMTTBL: ;Enemy Type Table
CB00             DW      0
CB02 00CF00CF    DW      TLINE,TLINE
CB06 00CF00CF    DW      TLINE,TLINE
CB0A 00CF00CF    DW      TLINE,TLINE
CB0E 00CF00CF    DW      TLINE,TLINE
CB12 00CF00CF    DW      TLINE,TLINE
CB16 00CF00CF    DW      TLINE,TLINE
CB1A 00CF00CF    DW      TLINE,TLINE
CB1E 00CF00CF    DW      TLINE,TLINE
CB22 00CF00CF    DW      TLINE,TLINE
CB26 00CF00CF    DW      TLINE,TLINE
CB2A 00CF00CF    DW      TLINE,TLINE
CB2E 00CF00CF    DW      TLINE,TLINE
CB32 00CF00CF    DW      TLINE,TLINE
CB36             EMSTBL:DS 54 ;Enemy score Table

;
0004             SKYP1: EQU 4      ;SKY Pattern 1
0020             EXPP1: EQU 32     ;EXPllosion Pattern 1
0026             LDEADP:EQU 38     ;Land DEAD Pattern
CB6C             PDBASE:DS 0C0H    ;Pattern Data BASE address

;
0010             EMWLEN:EQU 16     ;Enemy Work LENGTH
001E             EMVAL: EQU 30     ;Enemy VALue
CC2C             EMWORK:DS 400     ;Enemy WORK area
0004             EMBWLE:EQU 4      ;Enemy Bullet Work LENGTH
0028             EMBVAL:EQU 40     ;Enemy Bullet VALue
CE0C             EMBWOK:DS 160     ;Enemy Bullet WORK area

;
0003             MYBWLE:EQU 3      ;MY Bullet Work LENGTH
000C             MYBVAL:EQU 12     ;MY Bullet VALue
CEAC             MYBWOK:DS 36      ;MY Bullet WORK area

;
0030             REND: EQU 48      ;Right END
005C             DEND: EQU 92      ;Down END

;
CED0             MYLOC: DS 2       ;MY LOCation
CED2             MYRST: DS 1       ;MY ReST
CED3             SSKEY: DS 1       ;Space & Shift KEY

;
CED4             EDPO: DS 2        ;Enemy Data POinter
CED6             SCTOP: DS 2       ;SCroll TOP address
CED8             SCCT: DS 1        ;SCroll Counter

```

20110

```

CED9 00E0      ; SCENT: DW 0E000H ;SCroll data ENTry
CEDB 00E0      EMENT: DW 0E000H ;EneMy data ENTry

```

```

;
ORG 0CF00H

```

```

0000      ; STOP: EQU 0 ;STOP
0001      RR: EQU 1 ;Direction=1
0002      UR: EQU 2 ;Direction=2
0003      UU: EQU 3 ;Direction=3
0004      UL: EQU 4 ;Direction=4
0005      LL: EQU 5 ;Direction=5
0006      DL: EQU 6 ;Direction=6
0007      DD: EQU 7 ;Direction=7
0008      DR: EQU 8 ;Direction=8

```

```

0010      ; S1: EQU 10H ;Shot 1
0020      S2: EQU 20H ;Shot 2
0030      S3: EQU 30H ;Shot 3

```

```

0080      ; @END: EQU 80H ;@END command of QRL
0081      @JUMP: EQU 81H ;@JUMP command of QRL
0082      @IFZ: EQU 82H ;@IFZ command of QRL
0083      @IFC: EQU 83H ;@IFC command of QRL
0084      @CALL: EQU 84H ;@CALL command of QRL
0085      @FETCH: EQU 85H ;@FETCH command of QRL

```

```

CF00      ; TLINE: ;Test LINE
CF00 17      DB DD+S1
CF01 81      DB @JUMP
20410 CF02 00CF DW TLINE

```

234

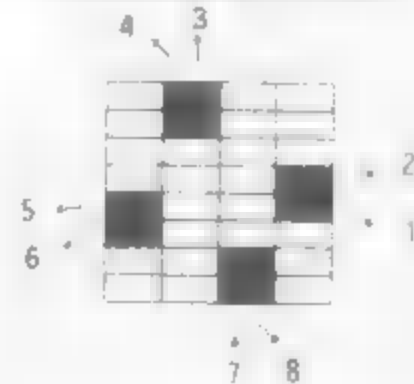
5

敵ワークエリアの内容

(1X+0)	FDH:地上敵爆発中
	FEH:空中敵爆発中
	FFH:敵出現中
	0:空き
(1X+1)	敵パターン番号
(1X+2)	X座標
(1X+3)	Y座標
(1X+4)	コマンド・ポイント(下位)
(1X+5)	コマンド・ポイント(上位)
(1X+6)	アップ・スコア(下位)
(1X+7)	アップ・スコア(上位)
(1X+8)	移動方向
(1X+9)	:
	敵のタイプにより、特殊
	な用事で使われること
	がある(List6-4)
(1X+15)	:

6

方向別弾発射位置



7

方向別弾発射の確率

方 向	1	2	3	4	5	6	7	8
フィルタ①	3/4	2/4	1/4	3/4	3/4	1	1	1
フィルタ②	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2
最終確率	3/8	2/8	1/8	2/8	3/8	1/2	1/2	1/2

4. スカイ・ブルーザー…Playing Game

会うは別れの始めなり。といいますが、マシン語ゲームの入門書として、ゼロからスタートをした本書も、事実上最後のプログラムを迎えることになりました。1章の最初のプログラムと比較すると、たった1冊の本でここまでくるのは、かなり無理があったような気もしますが、それを乗り越えて読破されてきたあなたの努力と根性には、心より敬意を表します。とにかく他人の作ったプログラムというものは、わかりにくくそして理解するのに骨が折れるものです。それは、たとえ同じ結果を求めるプログラムでも、作った人により考え方やアルゴリズムがまったく違うからです。ちょうど碁盤の目の一角から、線に沿って対角の頂点をめざす場合に、何通りもの道があるように、プログラムには絶対的な正解手順というものはありません。本書の中にも、不満を感じるルーチンがあったと思います。逆の見方をすれば、それがマシン語をマスターした何よりの証拠でもあるわけです。はるか遠くにあったゴールが、いつのまにかこんなにも近くに來ているのです。サァー、一気に、ゴールインしてしましましょう!!

さて、この最後のプログラムですが、マシン語プログラムというより、ほとんどが敵移動のためのQRLプログラムとなっています。また、ゲームとして最後の仕上げもしなければなりませんから、メイン・ルーチンの方も64KフルRAM・モードにしたり、パレットを変えたり、メッセージを入

れたり…と、これまでのテスト・プログラムに比べてかなり長くなっています。また、パターン・データ、マップ・データ、敵データも必要となります。そこで、プログラムの完成と同時に、ゲームとしても一応完成となるように、これらのデータの方から先に作成していくことにしましょう。

まずは、パターン・データからですが、このゲーム『スカイ・ブルーザー』で使用可能なパターン数は、敵が26種類、背景が52種類となっています。これ全部をデータ化するのは、QRLプログラムのことも考えると、もはやマシン語マスター用練習プログラムではなく、本物の商品作りになってしまいますから、ここでは、地上敵=3種類、空中敵=12種類、背景=26種類にしています。この他に自機、爆発シーン、残骸、そして弾のパターン・データも必要ですから、これでも結構多いと感じるかもしれません。しかし、商品に近づけるという目標のためには、この位は最低限用意しないと面白くありません。

パターン・データは背景がブルー面だけのデータ、それ以外はレッド面とグリーン面についてのデータ(データ・タイプは1)ですが、弾のデータ・タイプはこれまでのように2を選択してから、ブルー面を消してください。また、パターン作成時にはパレットをゲーム中と同じように変更しなければなりません。毎回変更するのは大変ですから、パターン・エディタのプログラム中に、次の行を追加しておくとおもしろくなります。

1025 COLOR = (0,1), (1,4), (3,2), (4,0), (5,0), (6,7)

作成されたパターン・データの転送先は、裏 RAM になります。ディスクの場合は、そのままのアドレスで自動的に裏 RAM のセーブ/ロードができますが、テープの場合は直接裏 RAM にセーブ/ロードすることはできません。そのため、データは C000H 番地から作成してセーブし、ロード後にモニタの M コマンド裏 RAM のアドレスへ

転送してください。なお、パターン番号 28H-2BH は「部分描き変えパターン」ですので、実際には縦 2 ドット分のデータがあればいいのですが、マップ・データ作成ツールで必要なため、このままのサイズで作成してください。

パターン・データの作成はカラーページのパターン②、③を見ながら、ハターン・エディ

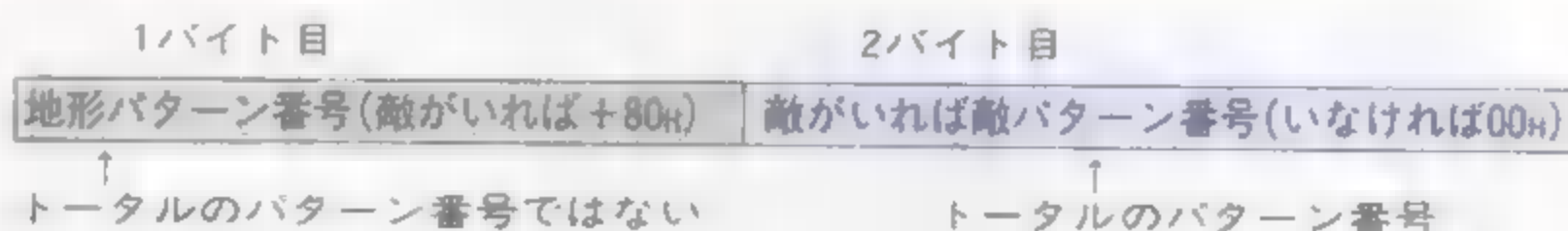
図 6

「Map Editor」で作製するデータの構造と、「Data Maker」によるデータ作成方法

1 「Map Editor」で作製するデータの構造

BFF0H.....カーソル X 座標
BFF1H.....カーソル Y 座標
BFF2~3H.....カーソルまでの総ライン数-1
BFF4~5H.....画面最下段までの総ライン数-1
BFF6~7H.....マップの総ライン数-1
BFFFH.....パターン番号
C000H.....

E5ADH(最大値) } マップの最下段から 2 バイト 1 組でデータが入って行く (スタート・アドレス C000H)



2 「Data Maker」でのデータ作成方法

1. 1000H 番地から 182 バイトを 03H で埋める (スクロール・エンドのつなぎ目となる)
2. 1. に続いて、「Map Editor」によるデータの最後から 1 バイト目のデータを順に入れていく.....マップ・データ作成
3. 6000H 番地より、「Map Editor」によるデータの先頭から 2 バイト目のデータ (00H 以外) を順に入れていく.....敵データ作成

4. アペンドをする場合

- アペンドする分だけ、作成されたマップ・データを転送し、そこにアペンドされるマップ・データを入れる
- 作成された敵データに続いて、敵データをアペンドする

5. データ作成終了の場合

- 作成されたマップ・データに続いて、182 バイトを 03H で埋める (スクロール・スタート時の初期画面用)
 - 6000H 番地以降に作られている敵データを、マップ・データの最後 (スクロールの先頭) とつなげるよう転送する
6. 1000H 番地から、敵データのエンドまでをセーブする
 7. セーブされたデータのスタート・アドレス (1000H 番地) を 0000H 番地に変更する (エンド・アドレスも、当然変更する)

タを使用して作ってください。次にマップ・データと敵データを作成しなくてはなりません。これはAppendix 2のマップ・エディタ「maped」とデータ・ジェネレータ「maker」という2つのツールを用いて作ります。

プログラムのマシン語部分については、紙面のつごうでBASICプログラム中にDATA文の形で書かれています。アセンブラのソースリストは、ディスク・アルバム10(発売予定)には付いておりますので、ぜひ見たい方はディスク・アルバムを購入してください。ただし、この2つのツールは両方共ディスク専用で、残念ながらテープで

の使用はできません。ごめんなさい。

エディット用のマップ・データは、実際にゲームで使用されるデータとは別の形で作られています。また、エディットできるマップの行数(縦のパターン総数)は、370行が最大となっています。しかし、実際のマップ・データ・エリア(0000H-5FFFH)には、この5倍位の長さのマップ・データを入れることができるので、長いマップは分割して作り、最後にまとめてアペンドするようになっています。例として、2つのマップをエディットしてから、アペンドするまでの作成手順を書いてみます。

1. 「How many files(0-15)?」には、1を入力する
2. パターン・データをロードする
3. 「maped」をロードする
4. 背景・敵をテン・キー及びアルファベット・キー(小文字)等を用いセットする
5. エディットしたデータをセーブ…ファイル名=Aとする
6. 4.の作業を繰り返す
7. エディットしたデータをセーブ…ファイル名=Bとする
8. 修正箇所があれば、A,B共に何度でもロードして(4)の作業をする
9. 「maker」をロードする
10. 1(GENERATE DATA)を押し、ロード・ファイル名=Aを入力する
11. 1(GENERATE DATA)を押し、ロード・ファイル名=Bを入力する
12. 2(END)を押し、セーブ・ファイル名を入力する…マップ・データ完成
13. SCENTとEMENTのアドレスを控える…メイン・プログラムに書き込む

以上が、マップ・データの作成方法ですが、ツールの操作についてはすべてメッセージで示されますので、ここで覚えるより実際にやってみるのが一番です。参考までに、マップ・エディタで作られるデータの構造と、データ・ジェネレータでのデータ作成方法を左ページに示しておきました。

最初はテスト用ですから、それほど長い

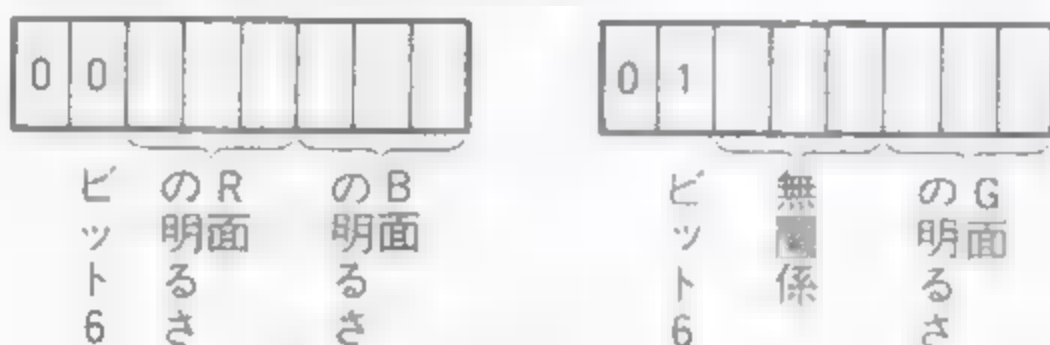
マップを作りませんが、これだけの敵と地形パターンを用意したのですから、アペンドしないまでも、せめて370行全部を使ったデータは欲しいものです。そして、マップ・データの作成が完了したら、残るはいよいよメイン・プログラムだけとなります。

メイン・プログラムを見てると、まずパレットの変更をしています。ここでめんど

图 7

V2 モードにおけるバレットの変更

出力ポートへ出力する値



各プレーン共、明るさ(0-7)のレベルとなる

うなのは SR の V2 モード時についてだけで、それ以外のモードでは出力ポート 54H-5BH が、単純にパレット番号の小さい方から並んでいるだけですから、プログラムを見ればその変更方法は容易に想像できると思います。本来、「SR の場合は V1 モードにしてください」と書いてしまえば、それ以上の説明は不要なのですが、せっかくですから V2 モードでのパレット変更方法も覚えてしまおう、というのがここに入れた理由です。V2 モードの場合は、カラー・パレットを 512 色中 8 色を選択するようになっていますが、これは各プレーン 8 レベルの発色ができるようになっているからです。そのため、1 プレーンの発光色を表現するのに 3 ビット必要になり、1 バイトでは同

時に3プレーンの表現ができなくなってしまうのです。そこで、各出力ポート共左図のようにして、3プレーンの色を表現するようにしています。

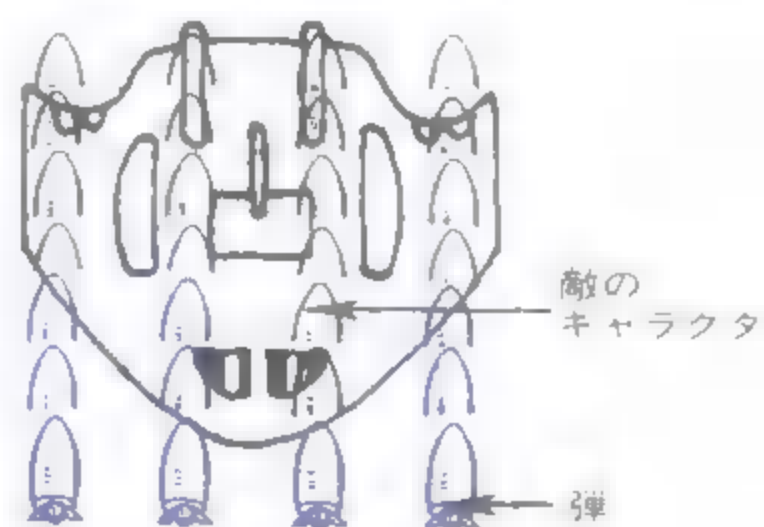
左図にあるように、ビット6の値によって選択するプレーンがブルー、レッド面と、グリーン面とに分けられますから、パレット変更は同じ出力ポートへ、2度違った値を出力しなければならないことになります。もちろん、パレットを変更しても、その値が変わらない場合(例えば、カラー番号=0を1にする場合、グリーン面の値は不変)には、必要な面についてだけ変更すればいいのです。

次に、64K フル RAM・モードの設定方法ですが、出力ポート 31H のビット 1 を立てることにより、裏 RAM と直接つながります。このポートの値は、CLS のルーチンでも使用しているように、E6C2H 番地にストアされています。

メイン・ループにおいては、敵の移動2回に対し、自機の弾が5回移動するようにしています。これは、当初1対2の割合にする予定でしたが、自機の弾だけより早く動くようにという要望が強かったため、仕方

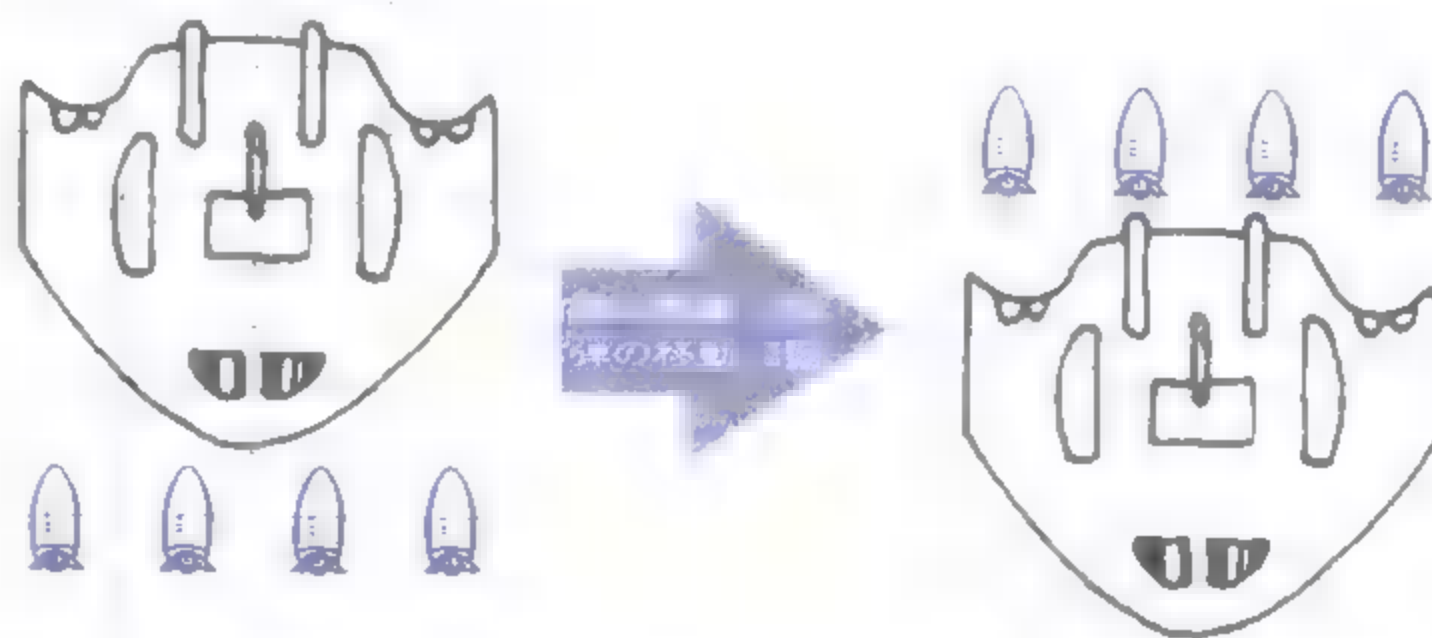
■ 判定の基準

弾のパターンが敵のパターンと完全に重なった時衝突とみなす



●判定モレが出る条件

最初の移動チェック

 8

なく変更したものです。仕方なくというのは、敵と自機の弾との衝突チェックは、敵移動後にだけ行なわれているため、図8のようなケースに素通り現象が起きてしまうからです。

これは確率的にはそれほど多く発生する

1. 判定の基準(List 6-3 の MYBCHK)を、もう1コマ下まで有効にする。ただし、衝突となった場合には、このままでは弾の残骸が半分残ることになるので、弾の消去が必要である。
2. 敵の移動1回に対し、自機の弾を3回移動する場合には、間に衝突判定のルーチンを追加する。

いずれにしても、それほどの問題ではありませんので、気になる方だけ変更を試みてください。また、個々の QRL の内容については、文章で説明するより画面でその動きを見た方がわかりやすいので、ここでは特に取り上げません。敵の動きは、自機を不死身にした上で、**HOME/CLR** を押しながらジックリと観察してください。不死身にする方法は、メイン・ループ中の MYCHK 後の2バイト(3ヶ所)を取るだけです。QRL によるプログラムを自分で作成

ものではないし、逆に弾と敵との高低差の違いによるズレがあると考えればそれも面白いということで、ここではそのままにしています。しかし、次のどちらかの方法により簡単に対処することができます。

してみたいという場合は、まず方眼紙に移動させたいラインを方向番号に合わせた移動距離で引いておきます。その後で、弾の発射やサブ・コマンドのことを考えながら、プログラムしていくと作りやすいと思います。

最後に、このリストで追加になったサブ・コマンドを、下の一覧表にまとめましたので、参考にしてください。

また、マップ・エディタを使って、自分のマップ・データを作ったら、その最後に表

表 4

サブ・コマンド

サブ・コマンド	内 容
POSCK (POSition Check)	自機と敵との Y 軸の差 $\geq 28H$ なら、キャリーフラグを立てて戻る
SETIX (SET IX register)	(IX + 次の1バイト値)に、その次の1バイト値を入れる
CTIX (Count IX)	(IX + 次の1バイト値)を-1し、 ゼロでなければキャリーフラグを立てて戻る
INTSC (INiTialize Sky C)	空中敵-C 専用のローカル・ワーク値設定 ……ワークエリア内容についてはプログラム参照
REVIVE (REVIVE enemy)	空中敵-C 専用の移動ルーチン ……弾に当たっても5回返は不死身で、SHOTAD を行う

使用例

α CALL	SETIX	<offset><value>	(IX + <offset>) ← <value>
α IFC	CTIX	<offset><NP>	(IX + <offset>)を-1し0であったら<NP>へジャンプ

示されたアドレスに変更してからアセンブルしてください。アセンブルが無事に済めば、長かった『スカイ・ブルーザー』のプログラムも、完成に大きく近づいたわけです。前回の2つのアセンブルしたプログラムをロードし、パターン・データ(6000H番地～)、マップ・データ(0000H番地～)、さらにはこれまでの2つのゲームに使用した数字・文字のデータもBA00H番地からへロードしてください。これで、すべての準備は完了となりました。

h)GC800

プログラムを走らせて、一発でうまく動作してくれた方は、もう我が世の春とばかりに遊んでしまってもいいでしょう。そうでない方は、再び春をめざしてツラ～イ世界、すなわちデバッグ作業へと戻らなければなりません。といっても、完動のリストが一応あるわけですから、春が来るのはもう時間の問題です。頑張ってください。

Quasi(グワシノ…ではありません…クワージです)大作の『スカイ・ブルーザー』、

いかがでしたか。コンストラクション機能をフルに使えば、かなり長く楽しめる反面、商品にするには今0.3歩の壁を感じる方もいるかもしれません。例えば、タイトルやデモ画面、ゲームにおける奥の深さ、意外性、パターンの総数、そしてサウンド…と、数えたらキリがありません。しかし、それらは世に出ている商品の価値を保ためにも、またあなた自身の創造性を養うためにも、「なくてよかった」と思って欲しいのです。本書にあるすべての作品に対し、大いに不満を感じてもらうことができたならば、正に本書の目的100%達成できたといっても過言ではないでしょう。要は、その不満をあなた自身の作品にぶつけければいいのです。そこには、単なるモノマネではない確固たる技術の進歩があるからです。どうか、あなた自身の工夫、改良によるスバラシイ作品を作られることを、心より期待します。間違っても、筆者への怒りの投書などという形で、不満の表現をすることだけはしないでください。ネッ!!

List 6-4 スカイ・ブルーザーの仕上げ

```

10000          ;***** List 6-4 *****
                ;
BD20          WAIT: EQU 0BD20H
BD2C          SETINT:EQU 0BD2CH
BD43          ORIINT:EQU 0BD43H
BD54          DISP: EQU 0BD54H
BD83          CLPTXY:EQU 0BD83H
BDCE          DISPLE:EQU 0BDCEH
BDF8          CLS: EQU 0BDF8H
BEA4          MAKESC:EQU 0BEA4H
BF20          MSGPRN:EQU 0BF20H
BF3D          DISPSC:EQU 0BF3DH
10120 BF6F          SCORE2:EQU 0BF6FH

```

← 作成したルーチンのアドレス

BF72	DUMMY: EQU	0BF72H
BF75	RND: EQU	0BF75H
BF88	RNDWOK: EQU	0BF88H
BF8A	MYMOVE: EQU	0BF8AH
C0A9	MYCHK: EQU	0C0A9H
C0FA	SSKCK: EQU	0C0FAH
C138	MYBMOV: EQU	0C138H
C16D	SCROLL: EQU	0C16DH
;		
C349	EMMVAL: EQU	0C349H
C39A	ENEMY: EQU	0C39AH
C554	EMDISP: EQU	0C554H
C58D	EMBMOV: EQU	0C58DH
C66E	ESHOT1: EQU	0C66EH
C6BF	DIRME: EQU	0C6BFH
C713	SWINGD: EQU	0C713H
C738	SAMDIR: EQU	0C738H
C73C	WHLR: EQU	0C73CH
C744	WHDU: EQU	0C744H
C74B	SHOTAD: EQU	0C74BH
C75B	LSHOT1: EQU	0C75BH
C763	LSHOT2: EQU	0C763H
C76B	LSHOT3: EQU	0C76BH

ORG 0C800H

C800	TEST: ;TEST	
C800 F3	DI	
C801 AF	XOR	A
C802 D351	OUT	(51H),A
C804 3100B6	LD	SP,0B600H

C807 3AD779	LD	A,(79D7H)
C80A FE34	CP	34H
C80C 3828	JR	C,PC88
C80E DB31	IN	A,(31H)
C810 E680	AND	80H
C812 2022	JR	NZ,PC88

C814 3E07	LD	A,7
C816 D354	OUT	(54H),A
C818 3E00	LD	A,0
C81A D355	OUT	(55H),A
C81C 3E47	LD	A,47H
C81E D355	OUT	(55H),A
C820 3E38	LD	A,38H
C822 D357	OUT	(57H),A
C824 3E40	LD	A,40H
C826 D358	OUT	(58H),A
C828 3E00	LD	A,0
C82A D359	OUT	(59H),A
C82C 3E40	LD	A,40H
C82E D359	OUT	(59H),A
C830 3E3F	LD	A,3FH
C832 D35A	OUT	(5AH),A
C834 1816	JR	PCEND

PC88: ;PC-8801 & v1 mode

C836 3E01	LD	A,1
C838 D354	OUT	(54H),A
C83A 3E04	LD	A,4
C83C D355	OUT	(55H),A
C83E 3E02	LD	A,2
C840 D357	OUT	(57H),A

1075C

10760	C842 3E00	LD	A,0	
	C844 D358	OUT	(58H),A	
	C846 D359	OUT	(59H),A	
	C848 3E07	LD	A,7	
	C84A D35A	OUT	(5AH),A	
	C84C			; PCEND: ;Pallet Change END
	C84C 3AC2E6	LD	A,(0E6C2H)	
	C84F F602	OR	2	64K フル RAM モード
	C851 32C2E6	LD	(0E6C2H),A	E6C 番地の第1ビットを立てておく(OUT
	C854 CDF8BD	CALL	CLS	(31H AはCLS内で実行している)
	C857 ED5F	LD	A,R	
	C859 3288BF	LD	(RNDWOK),A	R レジスタの値を乱数初期値にする
	C85C 2145CA	LD	HL,TITLE	
	C85F 011416	LD	BC,1614H	「SKY BRUISER」の表示
	C862 CD20BF	CALL	MSGPRN	
	C865 212ECA	LD	HL,PRESS	
	C868 01183F	LD	BC,3F18H	「PRESS RETURN KEY」の表示
	C86B CD20BF	CALL	MSGPRN	
	C86E			; TLOOP: ;Test LOOP
	C86E DB09	IN	A,(9)	
	C870 1F	RRA		が押されていなければ JEN
	C871 D295C9	JP	NC,PREND	
	C874 DB01	IN	A,(1)	
	C876 17	RLA		が押されていなければ JEN
	C877 38F5	JR	C,TLOOP	
	C879 CDF8BD	CALL	CLS	
	C87C CDA4BE	CALL	MAKESC	画面表示
	C87F 2AD9CE	LD	HL,(SCENT)	マッパ・ワーク・ポインタの初期化
	C882 22D6CE	LD	(SCTOP),HL	
	C885 AF	XOR	A	
	C886 32D8CE	LD	(SCCT),A	マッパ・ワーク・ポインタの初期化
	C889 2ADBCE	LD	HL,(EMENT)	敵・マッパ・ポインタの初期化
	C88C 2B	DEC	HL	(-1)は、敵出現ルーチンで+1後
	C88D 22D4CE	LD	(EDPO),HL	データポインタのため
	C890 3E05	LD	A,5	自機弾数の設定
	C892 32D2CE	LD	(MYRST),A	
	C895 210000	LD	HL,0	
	C898 226FBF	LD	(SCORE2),HL	スコアの初期化
	C89B 2270BF	LD	(SCORE2+1),HL	
	C89E 21ACCE	LD	HL,MYBWOK	
	C8A1 110300	LD	DE,MYBWLE	
	C8A4 060C	LD	B,MYBVAL	自機の弾のワークエリア初期化
	C8A6			(出現フラグをすべて0にする)
	C8A6 3600	LD	(HL),0	
	C8A8 19	ADD	HL,DE	
	C8A9 10FB	DJNZ	IMYBLP	
	C8AB 210CCE	LD	HL,EMBWK	
	C8AE 110400	LD	DE,EMBWL	
	C8B1 0628	LD	B,EMBVAL	敵の弾のワークエリア初期化
	C8B3			(出現フラグをすべて0にする)
	C8B3 3600	LD	(HL),0	
	C8B5 19	ADD	HL,DE	
	C8B6 10FB	DJNZ	IEMBLP	
	C8B8 212CCC	LD	HL,EMWOK	
11370	C8BB 111000	LD	DE,EMWLEN	

11380	C8BE 061E	LD B,EMVAL	
	C8C0	IEMLP: ;Initialize EneMy Loop	敵のワークエリア初期化 (出現フラグをすべて0にする)
	C8C0 3600	LD (HL),0	
	C8C2 19	ADD HL,DE	
	C8C3 10FB	DJNZ IEMLP	
		;	
	C8C5 013E06	LD BC,063EH	
	C8C8 213FCA	LD HL,SCORE	「SCORE」の表示
	C8CB CD20BF	CALL MSGPRN	
	C8CE 110000	LD DE,0	「000000」の表示
	C8D1 CD30BF	CALL DISPSC	
	C8D4 01470E	LD BC,0E47H	
	C8D7 2172BF	LD HL,DUMMY	タミー「100」の表示
	C8DA CD20BF	CALL MSGPRN	
	C8DD 013E18	LD BC,183EH	
	C8E0 AF	XOR A	
	C8E1 CD548D	CALL DISP	自機の表示(残数表示用)
	C8E4 3AD2CE	LD A,(MYRST)	
	C8E7 01451A	LD BC,1A45H	自機残数の表示
	C8EA CDCE8D	CALL DISPLE	
	C8ED CD2CB0	CALL SETINT	割り込みモードの設定
		;	
	C8F0	START: ;START	
	C8F0 21165B	LD HL,5816H	自機出現位置初期設定
	C8F3 22D0CE	LD (MYLOC),HL	
	C8F6 0610	LD B,10H	B・スター・■のフラッシュ回数
	C8F8	STLP: ;Start Loop	
	C8F8 C5	PUSH BC	
	C8F9 CDFAC0	CALL SSKCK	
	C8FC CD38C1	CALL MYBMOV	
	C8FF CDBDC5	CALL EMBMOV	
	C902 CD49C3	CALL EMMVAL	
	C905 CD6DC1	CALL SCROLL	
	C908 CDFAC0	CALL SSKCK	
	C90B CD38C1	CALL MYBMOV	
	C90E CD8ABF	CALL MYMOVE	
	C911 CD38C1	CALL MYBMOV	
	C914 CD208D	CALL WAIT	
	C917 CDFAC0	CALL SSKCK	
	C91A CD38C1	CALL MYBMOV	
	C91D CDBDC5	CALL EMBMOV	
	C920 CD49C3	CALL EMMVAL	
	C923 CD6DC1	CALL SCROLL	
	C926 CDFAC0	CALL SSKCK	
	C929 CD38C1	CALL MYBMOV	
	C92C CDBDC5	CALL EMBMOV	
	C92F CD8ABF	CALL MYMOVE	
	C932 ED4B00CE	LD BC,(MYLOC)	
	C936 211004	LD HL,410H	
	C939 CD83B0	CALL CLPTXY	
	C93C CD208D	CALL WAIT	
	C93F C1	POP BC	
	C940 10B6	DJNZ STLP	
		;	
	C942	MAIN: ;MAIN loop	
	C942 CDFAC0	CALL SSKCK	
	C945 CD38C1	CALL MYBMOV	
	C948 CDBDC5	CALL EMBMOV	
	C94B CD49C3	CALL EMMVAL	
	C94E CD6DC1	CALL SCROLL	
11380	C951 CDA9C0	CALL MYCHK	

```

C954 384F      JR    C,MYDEAD
C956 CDFAC0    CALL  SSKCK
C959 CD38C1    CALL  MYBMOV
C95C CD8ABF    CALL  MYMOVE
C95F CD38C1    CALL  MYBMOV
C962 CD20BD    CALL  WAIT
C965 CDFAC0    CALL  SSKCK
C968 CD38C1    CALL  MYBMOV
C96B CDBDC5    CALL  EMBMOV
C96E CD49C3    CALL  EMMVAL
C971 CD6DC1    CALL  SCROLL
C974 CDA9C0    CALL  MYCHK
C977 382C      JR    C,MYDEAD
C979 CDFAC0    CALL  SSKCK
C97C CD38C1    CALL  MYBMOV
C97F CDBDC5    CALL  EMBMOV
C982 CD8ABF    CALL  MYMOVE
C985 CDA9C0    CALL  MYCHK
C988 381B      JR    C,MYDEAD
C98A CD20BD    CALL  WAIT
C98D           M1: ;Ma:
C98D DB08      IN    A,(8)
C98F 1F        RRA
C990 3880      JR    C,MAIN
C992 1F        RRA
C993 38F8      JR    C,M1
C995           PREND: ;Program END
C995 CD43BD    CALL  ORIINT
C998 F3        DI
C999 3AC2E6    LD    A,(0E6C2H)
C99C E6F9      AND    0F9H
C99E 32C2E6    LD    (0E6C2H),A
C9A1 D331      OUT   (31H),A
C9A3 FB        EI
C9A4 FF        RST   38H

;
MYDEAD: ;MY DEAD
C9A5           MDLP: ;MyDead Loop
C9A5 0608      LD    B,8
C9A7 C5        PUSH  BC
C9A8 CD38C1    CALL  MYBMOV
C9AB CDBDC5    CALL  EMBMOV
C9AE CD49C3    CALL  EMMVAL
C9B1 CD6DC1    CALL  SCROLL
C9B4 CD38C1    CALL  MYBMOV
C9B7 3AC1E6    LD    A,(0E6C1H)
C9BA F620      OR     20H
C9BC D340      OUT   (40H),A
C9BE ED4BD0CE  LD    BC,(MYLOC)
C9C2 3E20      LD    A,EXPP1
C9C4 CD54BD    CALL  DISP
C9C7 3AC1E6    LD    A,(0E6C1H)
C9CA D340      OUT   (40H),A
C9CC CD38C1    CALL  MYBMOV
C9CF CD20BD    CALL  WAIT
C9D2 CD38C1    CALL  MYBMOV
C9D5 CDBDC5    CALL  EMBMOV
C9D8 CD49C3    CALL  EMMVAL
C9DB CD6DC1    CALL  SCROLL
C9DE CD38C1    CALL  MYBMOV
C9E1 CDBDC5    CALL  EMBMOV
C9E4 3AC1E6    LD    A,(0E6C1H)

```


13060	CB1C 20D53BD5		
	CB20 15CF15CF	DW	SKY1 ,SKY1 ,SKY1 ,SKY1 ,SKY1 ,SKY1 ,SKY1 ,SKY1
	CB24 15CF15CF		
	CB28 15CF15CF		
	CB2C 15CF15CF		
13070	CB30 15CF15CF	DW	SKY1 ,SKY1 ,SKY1
	CB34 15CF		
13080	CB36	EMSTBL:	略のスコアを示すデータ
13090	CB36 00000300	DW	0, 3, 6, 9, 1, 2, 3, 4
	CB3A 06000900		
	CB3E 01000200		
	CB42 03000400		
	CB46 05000600	DW	5, 6, 7, 8, 9, 10H, 11H, 12H
	CB4A 07000800		
	CB4E 09001000		
	CB52 11001200		
	CB56 00000000	DW	0, 0, 0, 0, 0, 0, 0, 0
	CB5A 00000000		
	CB5E 00000000		
	CB62 00000000		
13120	CB66 00000000	DW	0, 0, 0
	CB6A 0000		
	000F		
	0020		
	CB6C	SKYPC: EQU 15	
	CB6C 00608060	EXPP1: EQU 32	
	CB70 00618061	PDBASE:	
	CB74 00628062	DW	6000H, 6080H, 6100H, 6180H, 6200H, 6280H, 6300H, 6380H
	CB78 00638063		
	CB7C 00648064	DW	6400H, 6480H, 6500H, 6580H, 6600H, 6680H, 6700H, 6780H
	CB80 00658065		
	CB84 00668066		
	CB88 00678067		
	CB8C 00000000	DW	0, 0, 0, 0, 0, 0, 0, 0
	CB90 00000000		
	CB94 00000000		
	CB98 00000000		
	CB9C 00000000	DW	0, 0, 0, 0, 0, 0, 0, 0
	CBA0 00000000		
	CBA4 00000000		
	CBA8 00000000		
	CBAC 806E006F	DW	6E80H, 6F00H, 6E80H, 6F00H, 6E80H, 6F00H, 6F80H, 0
	CB80 806E006F		
	CB84 806E006F		
	CB88 806F0000		
	CB8C 00704070	DW	7000H, 7040H, 7080H, 70C0H, 7100H, 7140H, 7180H, 71C0H
	CBC0 8070C070		
	CBC4 00714071		
	CBC8 8071C071		
	CBCC 00724072	DW	7200H, 7240H, 7280H, 72C0H, 7300H, 7340H, 7380H, 73C0H
	CB00 8072C072		
	CB04 00734073		
	CB08 8073C073		
	CBDC 00744074	DW	7400H, 7440H, 7480H, 74C0H, 7500H, 7540H, 7580H, 75C0H
	CBE0 8074C074		
	CBE4 00754075		
	CBE8 8075C075		
13250	CBEC 00764076	DW	7600H, 7640H, 0, 0, 0, 0, 0, 0
	CBF0 00000000		
	CBF4 00000000		
	CBF8 00000000		
13260	CBFC 00000000	DW	0, 0, 0, 0, 0, 0, 0, 0

```

CC00 00000000
CC04 00000000
CC08 00000000
CC0C 00000000      DW  0  ,0  ,0  ,0  ,0  ,0  ,0  ,0
CC10 00000000
CC14 00000000
CC18 00000000
CC1C 00000000      DW  0  ,0  ,0  ,0  ,0  ,0  ,0  ,0
CC20 00000000
CC24 00000000
CC28 00000000

```

```

;
0010      EMWLEN:EQU 16
001E      EMVAL: EQU 30
CC2C      EMWORK:DS 480
;
0004      EMBWLE:EQU 4
0028      EMBVAL:EQU 40
CE0C      EMBWOK:DS 160
;
0003      MYBWLE:EQU 3
000C      MYBVAL:EQU 12
CEAC      MYBWOK:DS 36
;
CED0      MYLOC: DS 2
CED2      MYRST: DS 1
CED3      SSKEY: DS 1
;
CED4      EDPO:  DS 2
CED6      SCTOP: DS 2
CED8      SCCT:  DS 1
;
CED9 8013      SCENT: DW 138DH
CE0B 4314      EMENT: DW 1443H
;
;          ORG 0CF00H
;
0000      STOP:  EQU 0
0001      RR:    EQU 1
0002      UR:    EQU 2
0003      UU:    EQU 3
0004      UL:    EQU 4
0005      LL:    EQU 5
0006      DL:    EQU 6
0007      DD:    EQU 7
0008      DR:    EQU 8
;
0010      S1:    EQU 10H
0020      S2:    EQU 20H
0030      S3:    EQU 30H
;
0080      @END:  EQU 80H
0081      @JUMP: EQU 81H
0082      @IFZ:  EQU 82H
0083      @IFC:  EQU 83H
0084      @CALL: EQU 84H
0085      @FETCH:EQU 85H
;
CF00      LAND1:
CF00 84      DB  @CALL
CF01 5BC7      DW  LSHOT1
CF03 00      DB  STOP
13800 CF04 81      DB  @JUMP

```



```

CF05 00CF          DW    LAND1
;
CF07              ; LAND2:
CF07 84            DB    @CALL
CF08 63C7          DW    LSHOT2
CF0A 00            DB    STOP
CF0B 81            DB    @JUMP
CF0C 07CF          DW    LAND2
;
CF0E              ; LAND3:
CF0E 84            DB    @CALL
CF0F 6BC7          DW    LSHOT3
CF11 00            DB    STOP
CF12 81            DB    @JUMP
CF13 0ECF          DW    LAND3
;
CF15              ; SKY1:
CF15 07070707      DB    DD,DD,DD,DD,DD,DD,DD,DD
CF19 07070707      DB
CF1D 83            DB    @IFC
CF1E 3CC76ACF      DW    WHLR,S1R
CF22 16071706      DB    DL+S1,DD,DD+S1,DL,DD+S1,DL,DD+S1,DL
CF26 17061706      DB
CF2A 83            DB    @IFC
CF2B 3CC715CF      DW    WHLR,SKY1
CF2F 16070616      DB    DL+S1,DD,DL,DL+S1,LL,DL,DL+S1,DL,LL
CF33 05061606      DB
CF37 05            DB
CF38 15050515      DB    LL+S1,LL,LL,LL+S1
CF3C 83            DB    @IFC
CF3D 3CC763CF      DW    WHLR,S1L3
CF41 16071607      DB    DL+S1,DD,DL+S1,DD,DD+S1,DL,DD+S1,DD
CF45 17061707      DB
CF49 06070707      DB    DL,DD,DD,DD,DD,DD
CF4D 0707          DB
CF4F 83            DB    @IFC
CF50 3CC759CF      DW    WHLR,S1L1
CF54              ; S1L:
CF54 0617          DB    DL,DD+S1
CF56 81            DB    @JUMP
CF57 54CF          DW    S1L
CF59              ; S1L1:
CF59 080717        DB    DR,DD,DD+S1
CF5C              ; S1L2:
CF5C 07170716      DB    DD,DD+S1,DD,DL+S1
CF60 81            DB    @JUMP
CF61 63CF          DW    S1L3
CF63              ; S1L3:
CF63 05150514      DB    LL,LL+S1,LL,UL+S1
CF67 81            DB    @JUMP
CF68 63CF          DW    S1L3
CF6A              ; S1R:
CF6A 08170718      DB    DR,DD+S1,DD,DR+S1,DD,DR+S1,DD,DR+S1
CF6E 07180718      DB
CF72 83            DB    @IFC
CF73 3CC77ACF      DW    WHLR,S1R1
CF77 81            DB    @JUMP
CF78 15CF          DW    SKY1
CF7A              ; S1R1:
CF7A 08170818      DB    DR,DD+S1,DR,DR+S1,RR,DR+S1,DR,DR+S1
CF7E 01180818      DB
CF82 01010101      DB    RR,RR,RR,RR,RR
CF86 01            DB

```

CF87 83		DB	@IFC
CF88 3CC793CF		DW	WHLR, S1R3
CF8C	S1R2:		
CF8C 01110112		DB	RR, RR+S1, RR, UR+S1
CF90 81		DB	@JUMP
CF91 8CCF		DW	S1R2
CF93	S1R3:		
CF93 06071607		DB	DL, DD, DL+S1, DD, DD, DL+S1, DD, DD, DL+S1
CF97 07160707			
CF9B 16			
CF9C 07071707		DB	DD, DD, DD+S1, DD, DD
CFA0 07			
CFA1 83		DB	@IFC
CFA2 3CC7AFCF		DW	WHLR, S1R5
CFA6 060717		DB	DL, DD, DD+S1
CFA9	S1R4:		
CFA9 07170718		DB	DD, DD+S1, DD, DR+S1
CFAD A9CF		DW	S1R4
CFAF	S1R5:		
CFAF 0817		DB	DR, DD+S1
CFB1 81		DB	@JUMP
CFB2 54CF		DW	S1L
	;		
CFB4	SKY2:		
CFB4 07070706		DB	DD, DD, DD, DL, DD, DL, DL, DD, DD, DL, DL, DD
CFB8 07060607			
CFBC 07060607			
CFC0 83		DB	@IFC
CFC1 3CC7D2CF		DW	WHLR, S2R
CFC5 07060607		DB	DD, DL, DL, DD, DL, DL, DL, DL, DL, DL, DL, DL
CFC9 06060606			
CFCD 06060606			
CFD1 06			
CFD2	S2R:		
CFD2 33020302		DB	UU+S3, UR, UU, UR, UR, UU, UR, UU, UR, UU, UR, UU
CFD6 02030203			
CFDA 02030203			
CFDE 02020202		DB	UR, UR, UR, UR, UU, UR, UU, UR, UR, UR, UR, UR, UR
CFE2 03020302			
CFE6 02020202			
CFEA 02			
CFEB 83		DB	@IFC
CFEC 3CC7F3CF		DW	WHLR, S2R1
CFF0 81		DB	@JUMP
CFF1 06D0		DW	S2L
CFF3	S2R1:		
CFF3 02020202		DB	UR, UR, UR, UR, UR, UR, UR, RR, UR, UR, UR, RR, UR
CFF7 02020201			
CFFB 02020201			
CFFF 02			
D000 02020102		DB	UR, UR, RR, UR, UR, RR
D004 0201			
D006	S2L:		
D006 35050505		DB	LL+S3, LL, LL, LL, LL, DL, LL, LL, DL, DL, DL, DL
D00A 05060505			
D00E 06060606			
D012 06060606		DB	DL, DL, DL, DL
D016 83		DB	@IFC
D017 3CC729D0		DW	WHLR, S2R2
D01B 06060606		DB	DL, DL, DL, DL, DL, DD, DL, DL, DL, DD, DL, DD, DL
D01F 06070606			
D023 06070606			
D027 0706			

14760	D029	S2R2:	
14770	D029 33020302	DB	UU+S3,UR,UU,UR,UU,UR,UR,UU,UR,UU,UR,UR
	D020 03020203		
	D031 02030202		
14780	D035 03020202	DB	UU,UR,UR,UR,UR,UU,UR,UR
	D039 02030202		
14790	D03D 83	DB	@IFC
	D03E 3CC745D0	DW	WHLR,S2R3
	D042 81	DB	@JUMP
	D043 53D0	DW	S2L1
	D045	S2R3:	
14840	D045 02020202	DB	UR,UR,UR,UR,UR,UR,UR,RR,UR,UR,UR,RR,UR,UR
	D049 02020201		
	D040 02020201		
	D051 0202		
	D053	S2L1:	
	D053 36060607	DB	DL+S3,DL,DL,DD,DL,DL,DL,DD,DL,DD,DL,DD,DL
	D057 06060607		
	D05B 06070607		
	D05F 06		
4870	D060 83	DB	@IFC
	D061 3CC771D0	DW	WHLR,S2R4
4890	D065 07060707	DB	DD,DL,DD,DD,DL,DD,DD,DL,DD,DD,DL,DD
	D069 06070706		
	D06D 07070607		
	D071	S2R4:	
1491	D071 32020202	DB	UR+S3,UR,UR,UR,UR,UR,UR,UR,UR,UR,RR,UR,UR
	D075 02020202		
	D079 02020102		
	D07D 02		
	D07E 83	DB	@IFC
	D07F 3CC786D0	DW	WHLR,S2R5
	D083 81	DB	@JUMP
	D084 95D0	DW	S2L2
	D086	S2R5:	
	D086 02010202	DB	UR,RR,UR,UR,UR,RR,UR,UR,RR,UR,UR,RR,UR,RR,UR
	D08A 02010202		
	D08E 01020201		
	D092 020102		
	D095	S2L2:	
	D095 3606	DB	DL+S3,DL
	D097	S2L3:	
	D097 06060707	DB	DL,DL,DD,DD
	D09B 81	DB	@JUMP
	D09C 97D0	DW	S2L3
	D09E	;	
	D09E 07	SKY3:	
	D09F 83	DB	DD
	D0A0 0ED19ED0	DB	@IFC
	D0A4 83	DW	POSCK,SKY3
	D0A5 3CC7EBD0	DB	@IFC
	D0A9 83	DW	WHLR,S3R
	D0AA 3CC7D2D0	DB	@IFC
	D0AE	DW	WHLR,S3M
15140	D0AE 07070706	S3L:	
	D0B2 07060605	DB	DD,DD,DD,DL,DD,DL,DL,LL,LL,LL,LL
	D0B6 050505		
15150	D0B9	DB	@CALL
	D0BA 4BC7	DW	SHOTAD
	D0BC 0405	DB	UL,LL
	D0BE 84	DB	@CALL
	D0BF 4BC7	DW	SHOTAD


```

15200 D0C1 04040303      DB  UL,UL,UU,UU,UL,UU,UL,UU,UU,UU,UU,UL
      D0C5 04030403
      D0C9 03030303
      D0CD 04
15210 D0CE          S3U:      DB  UU
      D0CE 03          DB  @JUMP
      D0CF 81          DW  S3U
      D0D0 CED0
      D0D2          S3M:      DB  DD,DD,DD,DD,DD,DD,DD,STOP,STOP,STOP,STOP
      D0D2 07070707
      D0D6 07070700
      D0DA 000000
      D0DD 84          DB  @CALL
      D0DE 4BC7        DW  SHOTAD
      D0E0 0300        DB  UU,STOP
      D0E2 84          DB  @CALL
      D0E3 4BC7        DW  SHOTAD
      D0E5 030003      DB  UU,STOP,UU
      D0E8 81          DB  @JUMP
      D0E9 CED0        DW  S3U
      D0EB          S3R:      DB  DD,DD,DD,DR,DD,DR,DR,RR,RR,RR,RR
15360 D0EB 07070708
      D0EF 07080801
      D0F3 010101
      D0F6 84          DB  @CALL
      D0F7 4BC7        DW  SHOTAD
      D0F9 0201        DB  UR,RR
      D0FB 84          DB  @CALL
      D0FC 4BC7        DW  SHOTAD
      D0FE 02020303    DB  UR,UR,UU,UU,UR,UU,UR,UU,UU,UU,UU,UR
      D102 02030203
      D106 03030303
      D10A 02
130   D10B 81          DB  @JUMP
      D10C CED0        DW  S3U
      D10E          ; POSCK:      ; POSition Check
      D10E 3AD1CE      LD  A,(MYLOC+1)
      D111 DD9603      SUB (IX+3)
      D114 FE28        CP  28H
      D116 3F          CCF
      D117 C9          RET
      D118          ; SKY4:      ; 空中敵-4
1540   D118 07070707    DB  DD,DD,DD,DD,DD,DD,DD,DD,DD,DD,DD,DD
      D11C 07070707
      D120 07070707
      D124 07070707    DB  DD,DD,DD,DD,DD,DD,DD,DD,DD,DD,DD,DD
      D128 07070707
      D12C 07070707
15560 D130 07070707    DB  DD,DD,DD,DD,DD,DD,DD,DD,DD,DD,DD,DD
      D134 07070707
      D138 07070707
15570 D13C 83          DB  @IFC
      D13D 3CC786D1    DW  WHLR,S4R
      D141          S4L:
      D141 06          DB  DL
      D142          S4L1:
      D142 0504        DB  LL,UL
      D144 83          DB  @IFC
      D145 3CC74CD1    DW  WHLR,S4L2
      D149 81          DB  @JUMP
5660   D14A 42D1        DW  S4L1

```

D14C		S4L2:	
D14C 24232423	DB		UL+S2,UU+S2,UL+S2,UU+S2,UL+S2,UU+S2
D150 2423			
D152 23242323	DB		UU+S2,UL+S2,UU+S2,UU+S2,UU+S2,UU+S2
D156 2323			
D158 23232323	DB		UU+S2,UU+S2,UU+S2,UU+S2,UU+S1,UU+S1
D15C 1313			
D15E 13131313	DB		UU+S1,UU+S1,UU+S1,UU+S1,UU+S1,UU+S1
D162 1313			
D164 13131313	DB		UU+S1,UU+S1,UU+S1,UU+S1,UU+S1,UU+S1
D168 1313			
D16A 13130303	DB		UU+S1,UU+S1,UU,UU,UR,UU,UU,UU,UU,UU
D16E 02030303			
D172 0303			
D174 0302	DB		UU,UR
D176 02030303	DB		UR,UU,UU,UU,UU,UU,UU,UU,UU,UU,UU
D17A 03030303			
D17E 03030303			
D182		S4L3:	
D182 03	DB		UU
D183 81	DB		@JUMP
D184 82D1	DW		S4L3
D186		S4R:	
D186 08	DB		DR
D187		S4R1:	
D187 0102	DB		RR,UR
D189 83	DB		@IFC
D18A 3CC787D1	DW		WHLR,S4R1
D18E		S4R2:	
D18E 24232223	DB		UL+S2,UU+S2,UR+S2,UU+S2,UR+S2,UU+S2
D192 2223			
D194 23232323	DB		UU+S2,UU+S2,UU+S2,UU+S2,UU+S2,UU+S2
D198 2323			
D19A 23232323	DB		UU+S2,UU+S2,UU+S2,UU+S2,UU+S1,UU+S1
D19E 1313			
D1A0 13131313	DB		UU+S1,UU+S1,UU+S1,UU+S1,UU+S1,UU+S1
D1A4 1313			
D1A6 13131313	DB		UU+S1,UU+S1,UU+S1,UU+S1,UU+S1,UU+S1
D1AA 1313			
D1AC 13130303	DB		UU+S1,UU+S1,UU,UU,UL,UU,UU,UU,UU,UU
D1B0 04030303			
D1B4 0303			
D1B6 04030303	DB		UL,UU,UU,UU,UU,UU,UU,UU,UU,UU,UU
D1BA 03030303			
D1BE 03030303			
D1C2 0304	DB		UU,UL
D1C4 81	DB		@JUMP
D1C5 82D1	DW		S4L3
D1C7		i	
D1C7 07070707		SKY5:	
D1CB 07070707	DB		DD,DD,DD,DD,DD,DD,DD,DD,DD,DD,DD,DD
D1CF 07070707			
D1D3 07070707	DB		DD,DD,DD,DD
D1D7 83	DB		@IFC
D1D8 44C7ECD1	DW		WHDU,S51
D1DC 07070707	DB		DD,DD,DD,DD,DD,DD,DD,DD,DD,DD,DD,DD
D1E0 07070707			
D1E4 07070707			
D1E8 07070707	DB		DD,DD,DD,DD
D1EC		S51:	
D1EC 08070708	DB		DR,DD,DD,DR,DR,DR,RR
D1F0 080801			

D1F3		S52:	
D1F3 0201	DB	UR,RR	
D1F5 83	DB	@IFC	
D1F6 3CC7F3D1	DW	WHLR,S52	
D1FA 02010202	DB	UR,RR,UR,UR,UR,RR,UR,UU,UR,UU,UR,UU	
D1FE 02010203			
D202 02030203			
D206 03020303	DB	UU,UR,UU,UU,UU,UU,UR,UU,UU,UU,UU,UU	
D20A 03030203			
D20E 03030303			
D212 04030303	DB	UL,UU,UU,UU,UL,UL,UL,UL,UL,UL,LL,UL	
D216 04040404			
D21A 04040504			
D21E 05050607	DB	LL,LL,DL,DD,DL,DD,DD,DD,DD,DR,DD,RR	
D222 06070707			
D226 07080701			
D22A 01020102	DB	RR,UR,RR,UR,UR,UR,UR,UU,UU,UU,UU,UR	
D22E 02020203			
D232 03030302			
D236 03030303	DB	UU,UU,UU,UU,UU,UL,UU,UU,UL,UL,UL,LL	
D23A 03040303			
D23E 04040405			
D242 04060606	DB	UL,DL,DL,DL,DD,DD,DD,DR,RR,RR,UR,UR	
D246 07070708			
D24A 01010202			
D24E 02030303	DB	UR,UU,UU,UU,UU,UU,UU,UU,UL,UL,LL,DL	
D252 03030303			
D256 04040506			
D25A 07080102	DB	DD,DR,RR,UR,UU,UU,UL,LL	
D25E 03030405			
D262 00	DB	STOP	
D263 84	DB	@CALL	
D264 4BC7	DW	SHOTAD	
D266 00	DB	STOP	
D267 84	DB	@CALL	
D268 4BC7	DW	SHOTAD	
D26A 00	DB	STOP	
D26B 84	DB	@CALL	
D26C 4BC7	DW	SHOTAD	
D26E 80	DB	@END	
D26F		; SKY6:	
D26F 0717	DB	DD,DD+S1	
D271 83	DB	@IFC	
D272 44C779D2	DW	WHDU,S6LR	
D276 81	DB	@JUMP	
D277 6FD2	DW	SKY6	
D279		S6LR:	
D279 83	DB	@IFC	
D27A 3CC78BD2	DW	WHLR,S6R	
D27E 82	DB	@IFZ	
D27F 3CC76FD2	DW	WHLR,SKY6	
D283		S6L:	
D283 070706	DB	DD,DD,DL	
D286		S6L1:	
D286 1505	DB	LL+S1,LL	
D288 81	DB	@JUMP	
D289 86D2	DW	S6L1	
D28B		S6R:	
D28B 070708	DB	DD,DD,DR	
D28E		S6R1:	
D28E 1101	DB	RR+S1,RR	
D290 81	DB	@JUMP	

16530	D291 8ED2		DW	S6R1	
	D293	;			
	D293 84	SKY7:	DB	@CALL	全中継-7
	D294 B8D2		DW	SETIX	
	D296 0A08		DB	10,8	
	D298	S7LP:			
	D298 84		DB	@CALL	
	D299 B8D2		DW	SETIX	
	D29B 0B10		DB	11,16	
	D29D 85		DB	@FETCH	
	D29E BFC6		DW	DIRME	
	D2A0	S7LP1:			
	D2A0 84		DB	@CALL	
	D2A1 6EC6		DW	ESHOT1	
	D2A3 85		DB	@FETCH	
	D2A4 38C7		DW	SAMDIR	
	D2A6 83		DB	@IFC	
	D2A7 C6D2		DW	CTIX	
	D2A9 08		DB	11	
	D2AA A0D2		DW	S7LP1	
	D2AC 83		DB	@IFC	
	D2AD C6D2		DW	CTIX	
	D2AF 0A		DB	10	
	D2B0 98D2		DW	S7LP	
	D2B2	S7LP2:			
	D2B2 85		DB	@FETCH	
	D2B3 38C7		DW	SAMDIR	
	D2B5 81		DB	@JUMP	
	D2B6 B2D2		DW	S7LP2	
	D2B8	;			
	D2B8 E1	SETIX:	POP	HL	;SET IX register
	D2B9 D1		POP	DE	HL ← リターン・アドレス
	D2BA 13		INC	DE	DE ← コマント・ポインタ
	D2BB 1A		LD	A,(DE)	A ← 次ポインタの値
	D2BC 32C3D2		LD	(SX1+2),A	SX1のIX+0がIX+Aとなる
	D2BF 13		INC	DE	
	D2C0 1A		LD	A,(DE)	A ← 次ポインタの値
	D2C1	SX1:			
	D2C1 DD7700		LD	(IX+0),A	;SetIX 1
	D2C4 D5		PUSH	DE	(IX+0)にAの値が設定される
	D2C5 E9		JP	(HL)	スタックにコマント・ポインタ+2されている値へ戻す
	D2C6	;			
	D2C6 E1	CTIX:	POP	HL	;Count IX
	D2C7 D1		POP	DE	HL ← リターン・アドレス
	D2C8 13		INC	DE	DE ← コマント・ポインタ
	D2C9 1A		LD	A,(DE)	A ← 次ポインタの値
	D2CA 32CFD2		LD	(CX1+2),A	CX1のIX+0がIX+Aとなる
	D2CD	CX1:			
	D2CD DD3500		DEC	(IX+0)	;CtiX 1
	D2D0 2803		JR	Z,CX2	(IX+0)の値を-1し、ゼロでなければキャリーフラグを立てる。ゼロの時はCX2へ
	D2D2 37		SCF		
	D2D3 D5		PUSH	DE	スタックにコマント・ポインタ
	D2D4 E9		JP	(HL)	+1されている)へ戻す
	D2D5	CX2:			
	D2D5 B7		OR	A	;CtiX 2
	D2D6 D5		PUSH	DE	キャリーフラグのリセット
	D2D7 E9		JP	(HL)	スタックにコマント・ポインタ(1+されている)へ戻す
	D2D8	;			
	D2D8 17161717	SKY8:	DB	DD+S1,DL+S1,DD+S1,DD+S1,DL+S1,DD+S1	全中継-A
17150					

	D2DC	1617			
17160	D2DE	16171716	DB	DL+S1,DD+S1,DD+S1,DL+S1,DD+S1,DL+S1	
	D2E2	1716			
	D2E4	17171716	DB	DD+S1,DD+S1,DD+S1,DL+S1,DD+S1,DD+S1	
	D2E8	1717			
17180	D2EA	16171717	DB	DL+S1,DD+S1,DD+S1,DD+S1,DL+S1,DD+S1	
	D2EE	1617			
17190	D2F0	07070706	DB	DD,DD,DD,DL,DD,DD,DD,DD	
	D2F4	07070707			
17200	D2F8	06070707	DB	DL,DD,DD,DD,DD,DL,DD,DD	
	D2FC	07060707			
17210	D300	07070707	DB	DD,DD,DD,DD,DD,DD	
	D304	0707			
17220	D306	83	DB	@IFC	
	D307	3CC784D3	DW	WHLR,S8R	
	D308		S8L:		
	D308	84	DB	@CALL	
	D30C	B8D2	DW	SETIX	
	D30E	0A20	DB	10,32	
	D310		S8L1:		
	D310	13	DB	UU+S1	
	D311	83	DB	@IFC	
	D312	C6D2	DW	CTIX	
	D314	0A	DB	10	
	D315	10D3	DW	S8L1	
17340	D317	04030303	DB	UL,UU,UU,UU,UU,UU,UU,UU	
	D318	03030303			
17350	D31F	03030303	DB	UU,UU,UU,UU,UU,UU,UL,UU	
	D323	03030403			
17360	D327	03030303	DB	UU,UU,UU,UU,UU,UU,UU,UU	
	D32B	03030303			
17370	D32F	03030303	DB	UU,UU,UU,UU,UL,UU,UU,UU	
	D333	04030303			
17380	D337	03030303	DB	UU,UU,UU,UU,UU,UU,UR,UU	
	D33B	03030203			
17390	D33F	03030303	DB	UU,UU,UU,UU,UR,UU,UU,UU	
	D343	02030303			
17400	D347	02030303	DB	UR,UU,UU,UU,UR,UU,UU,UU	
	D34B	02030303			
17410	D34F	03030303	DB	UU,UU,UU,UU,DD,DL,DD,DD	
	D353	07060707			
17420	D357	06070607	DB	DL,DD,DL,DD	
17430	D35B	17161716	DB	DD+S1,DL+S1,DD+S1,DL+S1,DD+S1,DD+S1	
	D35F	1717			
17440	D361	17161717	DB	DD+S1,DL+S1,DD+S1,DD+S1,DL+S1,DD+S1	
	D365	1617			
17450	D367	17171617	DB	DD+S1,DD+S1,DL+S1,DD+S1,DD+S1,DD	
	D36B	1707			
17460	D36D	17061707	DB	DD+S1,DL,DD+S1,DD,DD+S1,DD	
	D371	1707			
17470	D373	06070707	DB	DL,DD,DD,DD,DD,DL,DD,DD	
	D377	07060707			
17480	D37B	07070707	DB	DD,DD,DD,DD,DD,DD	
	D37F	0707			
17490	D381	81	DB	@JUMP	
	D382	0BD3	DW	S8L	
	D384		S8R:		
	D384	23222323	DB	UU+S2,UR+S2,UU+S2,UU+S2	
	D388	23232323	DB	UU+S2,UU+S2,UU+S2,UU+S2	
17540	D38C	02030303	DB	UR,UU,UU,UU,UU,UR,UU,UU	
	D390	03020303			
17550	D394	03030303	DB	UU,UU,UU,UU,UU,UR,UU,UU	
	D398	03020303			

D39C	03030203	DB	UU, UU, UR, UU, UU, UU, UU, UU
D3A0	03030303		
D3A4	03030303	DB	UU, UU, UU, UU, UR, UU, UU, UU
D3A8	02030303		
D3AC	03030303	DB	UU, UU, UU, UU, UR, UU, UU, UR
D3B0	02030302		
D3B4	03030303	DB	UU, UU, UU, UU, UR, UU, UU, UU
D3B8	02030303		
D3BC	02030303	DB	UR, UU, UU, UU, UR, UU, UU, UR
D3C0	02030302		
D3C4	03030203	DB	UU, UU, UR, UU, UU, UU, UR, UU
D3C8	03030203		
D3CC	03030203	DB	UU, UU, UR, UU, UU, UR, UU, UU
D3D0	03020303		
D3D4	02030302	DB	UR, UU, UU, UR, UU, UR, UU, UU
D3D8	03020303		
D3DC	02030203	DB	UR, UU, UR, UU, DD, DL, DD, DD
D3E0	07060707		
D3E4	06070607	DB	DL, DD, DL, DD, DD+S1, DL+S1
D3E8	1716		
D3EA	17161717	DB	DD+S1, DL+S1, DD+S1, DD+S1
D3EE	17161707	DB	DD+S1, DL+S1, DD+S1, DD, DL+S1
D3F2	16		
D3F3	07170716	DB	DD, DD+S1, DD, DL+S1, DD, DD+S1
D3F7	0717		
D3F9	07070617	DB	DD, DD, DL, DD+S1, DD, DD, DD
D3FD	070707		
D400	16070707	DB	DL+S1, DD, DD, DD, DD, DL, DD
D404	070607		
D407	07170707	DB	DD, DD+S1, DD, DD, DD, DD, DD
D40B	070707		
D40E	81	DB	@JUMP
D40F	84D3	DW	S8R
;			
D411		SKY9:	
D411	84	DB	@CALL
D412	B8D2	DW	SETIX
D414	0A12	DB	10, 18
D416		S9LP:	
D416	85	DB	@FETCH
D417	13C7	DW	SWINGD
D419	84	DB	@CALL
D41A	B8D2	DW	SETIX
D41C	0B08	DB	11, 8
D41E		S9LP1:	
D41E	84	DB	@CALL
D41F	6EC6	DW	ESHOT1
D421	85	DB	@FETCH
D422	38C7	DW	SAMDIR
D424	83	DB	@IFC
D425	C6D2	DW	CTIX
D427	0B	DB	11
D428	1ED4	DW	S9LP1
D42A	83	DB	@IFC
D42B	C6D2	DW	CTIX
D42D	0A	DB	10
D42E	16D4	DW	S9LP
D430		S9LP2:	
D430	85	DB	@FETCH
D431	38C7	DW	SAMDIR
D433	81	DB	@JUMP
D434	30D4	DW	S9LP2

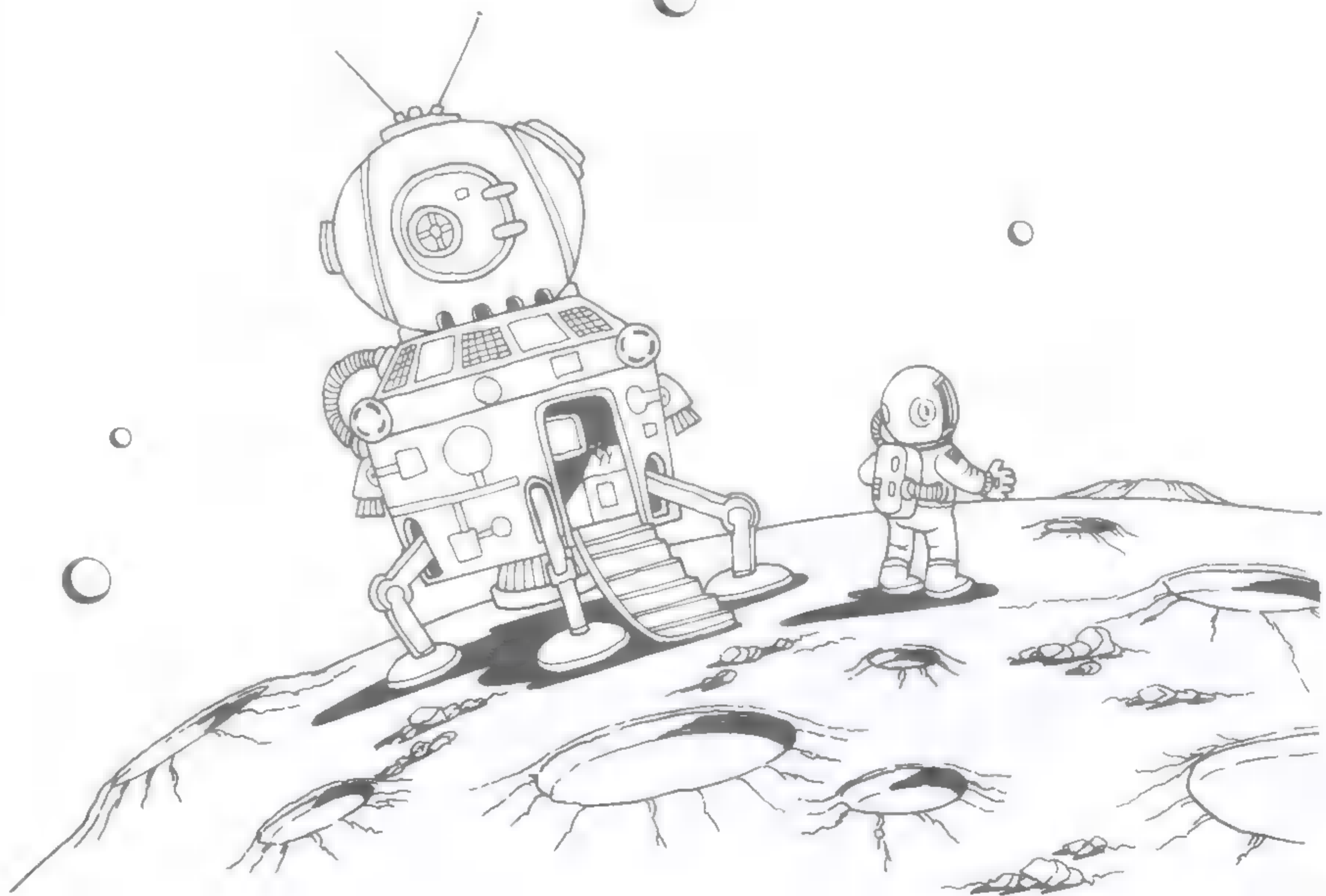
	D436	SKYA:	DB	LL, DL, DD, DR, DD, DL, DL, DD
	D436 05060708			
	D43A 07060607		DB	DD, DR, DD, DD, DL, DL, DD
	D43E 07080707			
	D442 060607			
	D445 83		DB	@IFC
	D446 3CC7DCD4		DW	WHLR, SAR
	D44A 82		DB	@IFZ
	D44B 3CC7DCD4		DW	WHLR, SAR
	D44F	SAL:		
	D44F 05060705		DB	LL, DL, DD, LL, LL, UL, LL
	D453 050405			
	D456 83		DB	@IFC
	D457 44C7A2D4		DW	WHOU, SAL7
	D45B	SAL1:		
18160	D45B 16171716		DB	DL+S1, DD+S1, DD+S1, DL+S1, LL+S1, UL+S1
	D45F 1514			
	D461 14141617		DB	UL+S1, UL+S1, DL+S1, DD+S1, DD+S1, DD+S1
	D465 1717			
18180	D467 16151415		DB	DL+S1, LL+S1, UL+S1, LL+S1, UL+S1, DD+S1
	D46B 1417			
18190	D46D 17171716		DB	DD+S1, DD+S1, DD+S1, DL+S1
	D471 83		DB	@IFC
	D472 44C788D4		DW	WHOU, SAL4
	D476	SAL2:		
823	D476 06040504		DB	DL, UL, LL, UL, UL, DD, DD, DD
	D47A 04070707			
	D47E 07060405		DB	DD, DL, UL, LL, UL, LL, DD, DD
	D482 04050707			
18251	D486	SAL3:		
	D486 0705		DB	DD, LL
	D488 81		DB	@JUMP
	D489 86D4		DW	SAL3
	D48B	SAL4:		
8301	D48B 08070601		DB	DR, DD, DL, RR, RR, DD, DD, RR
	D48F 01070701			
18311	D493 01010707		DB	RR, RR, DD, DD
	D497	SAL5:		
8330	D497 07060706		DB	DD, DL, DD, DL, DD
	D49B 07			
18340	D49C	SAL6:		
	D49C 060707		DB	DL, DD, DD
	D49F 81		DB	@JUMP
	D4A0 9CD4		DW	SAL6
	D4A2	SAL7:		
18391	D4A2 14141313		DB	UL+S1, UL+S1, UU+S1, UU+S1, UL+S1, DL+S1
	D4A6 1416			
18400	D4A8 15141413		DB	LL+S1, UL+S1, UL+S1, UU+S1, UU+S1, UL+S1
	D4AC 1314			
18411	D4AE 13141415		DB	UU+S1, UL+S1, UL+S1, LL+S1, UL+S1, UU+S1
	D4B2 1413			
1842	D4B4 13131313		DB	UU+S1, UU+S1, UU+S1, UU+S1, UU+S1, LL+S1
	D4B8 1315			
18430	D4BA 15141414		DB	LL+S1, UL+S1, UL+S1, UL+S1, UU+S1, UU+S1
	D4BE 1313			
	D4C0 24131313		DB	UL+S2, UU+S1, UU+S1, UU+S1, UU+S1, UU+S1
	D4C4 1313			
	D4C6 13131515		DB	UU+S1, UU+S1, LL+S1, LL+S1, LL+S1, UL+S1
	D4CA 1514			
18461	D4CC 14131213		DB	UL+S1, UU+S1, UR+S1, UU+S1, UU+S1, UR+S1
	D4D0 1312			
18470	D4D2 13131515		DB	UU+S1, UU+S1, LL+S1, LL+S1
18480	D4D6	SAL8:		

	D4D6 141514		DB	UL+S1,LL+S1,UL+S1
	D4D9 81		DB	@JUMP
	D4DA 0604		DW	SAL8
	D4DC	SAR:		
18530	D4DC 01010706		DB	RR,RR,DD,DL,DD,DL,DD,LL
	D4E0 07060705			
18540	D4E4 0505		DB	LL,LL
	D4E6 83		DB	@IFC
	D4E7 3CC719D5		DW	WHLR,SAR5
	D4EB	SAR1:		
18580	D4E8 16171616		DB	DL+S1,DD+S1,DL+S1,DL+S1,DD+S1,DD+S1
	D4EF 1717			
18590	D4F1 18171717		DB	DR+S1,DD+S1,DD+S1,DD+S1,DD+S1,DD+S1
	D4F5 1717			
18600	D4F7 83		DB	@IFC
	D4F8 3CC71305		DW	WHLR,SAR3
	D4FC	SAR2:		
	D4FC 82		DB	@IFZ
	D4FD 3CC71305		DW	WHLR,SAR3
18650	D501 05040607		DB	LL,UL,DL,DD,DL,LL,UL,LL
	D505 06050405			
18660	D509 04050606		DB	UL,LL,DL,DL,LL,UL,UL
	D50D 050404			
18670	D510 81		DB	@JUMP
	D511 97D4		DW	SAL5
	D513	SAR3:		
	D513 0708		DB	DD,DR
	D515	SAR4:		
	D515 07		DB	DD
	D516 81		DB	@JUMP
	D517 15D5		DW	SAR4
	D519	SAR5:		
18760	D519 11111212		DB	RR+S1,RR+S1,UR+S1,UR+S1,RR+S1,DR+S1
	D51D 1118			
18770	D51F 17111112		DB	DD+S1,RR+S1,RR+S1,UR+S1,RR+S1,RR+S1
	D523 1111			
18780	D525 18181818		DB	DR+S1,DR+S1,DR+S1,DR+S1
	D529	SAR6:		
	D529 11		DB	RR+S1
	D52A 81		DB	@JUMP
	D52B 29D5		DW	SAR6
	D52D	i SKYB:		
	D52D 84		DB	@CALL
	D52E 4BC7		DW	SHOTAD
18870	D530 00000000		DB	STOP,STOP,STOP,STOP,STOP,STOP,STOP,STOP
	D534 00000000			
18880	D538 81		DB	@JUMP
	D539 2DD5		DW	SKY8
	D53B	i SKYC:		
	D53B 84		DB	@CALL
	D53C 9BD5		DW	INTSC
	D53E	SCLP:		
	D53E 84		DB	@CALL
	D53F ADD5		DW	REVIVE
	D541 81		DB	@JUMP
	D542 3ED5		DW	SCLP
	D544	SC1:		
19000	D544 06060607		DB	DL,DL,DL,DD,DD,DD,DL,DL,DL
	D548 07070606			
	D54C 06			
19010	D54D	SC2:		

9020	0540 07080808	DB	DD,DR,DR,DR,RR,RR,UR,RR,UR,RR,UR,UR	
	0551 01010201			
	0555 02010202			
	0559 02010202	DB	UR,RR,UR,UR,RR,UR,UR,UR,UR,UR,RR,UR	
	055D 01020202			
	0561 02020102	DB	UR,UR,UR,UR,UR,UR,UU,UR,UR,UU,UR,UU	
	0565 02020202			
	0569 02020302			
90F	056D 02030203	DB	UU,UU,UU,UL,UU,UL,LL,UL,LL,UL,LL,UL	
	0571 03030304			
	0575 03040504			
	0579 05040504			
	057D 05040505	DB	LL,UL,LL,LL,UL,LL,LL,LL,UL,LL,LL,LL	
	0581 04050505			
	0585 04050505			
907	0589 05050505	DB	LL,LL,LL,LL,LL,LL,DL,LL,DL,DL,DL,DL	
	058D 05050605			
	0591 06060606			
908	0595 070607	DB	DD,DL,DD	
	0598 81	DB	@JUMP	
	0599 40D5	DW	SC2	
	0598			
	059B DD360A04	LD	(IX+10),4	IX+10 = 4 (復活回数 0+5 されるので、 実質 5 回)
	059F 2144D5	LD	HL,SC1	IX+10 = 2144D5 (実際のリマ ンタル設定)
	05A2 DD750C	LD	(IX+12),L	IX+12 = DD750C (実際のリマ ンタル設定)
	05A5 DD740D	LD	(IX+13),H	IX+13 = DD740D (実際のリマ ンタル設定)
	05A8 DD360E0F	LD	(IX+14),SKYPC	IX+14 = DD360E0F (実際のリマ ンタル設定)
	05AC C9	RET		
	05AD	REVIVE: ;REVIVE enemy		
	05AD D1	POP	DE	DE (使用しない)
	05AE DD6E0C	LD	L,(IX+12)	IX+12 = DD6E0C (実際のリマ ンタル設定)
	05B1 DD660D	LD	H,(IX+13)	IX+13 = DD660D (実際のリマ ンタル設定)
	05B4 DD7504	LD	(IX+4),L	IX+4 = DD7504 (実際のリマ ンタル設定)
	05B7 DD7405	LD	(IX+5),H	IX+5 = DD7405 (実際のリマ ンタル設定)
	05BA 3E0E	LD	A,14	IX+14 = 3E0E (実際のリマ ンタル設定)
	05BC 3261C5	LD	(EMDISP+13),A	IX+13 = 3261C5 (実際のリマ ンタル設定)
	05BF CD9AC3	CALL	ENEMY	IX+13 = CD9AC3 (実際のリマ ンタル設定)
	05C2 3E01	LD	A,1	IX+1 = 3E01 (実際のリマ ンタル設定)
	05C4 3261C5	LD	(EMDISP+13),A	IX+13 = 3261C5 (実際のリマ ンタル設定)
	05C7 DD7E00	LD	A,(IX+0)	IX+0 = DD7E00 (実際のリマ ンタル設定)
	05CA B7	OR	A	IX+0 = B7 (実際のリマ ンタル設定)
	05CB 2813	JR	Z,RVEND	IX+0 = 2813 (実際のリマ ンタル設定)
	05CD 3C	INC	A	IX+0 = 3C (実際のリマ ンタル設定)
	05CE 2812	JR	Z,RV1	IX+0 = 2812 (実際のリマ ンタル設定)
	05D0 DD3600FF	LD	(IX+0),0FFH	IX+0 = DD3600FF (実際のリマ ンタル設定)
	05D4 DD36010F	LD	(IX+1),SKYPC	IX+1 = DD36010F (実際のリマ ンタル設定)
	05D8 CD4BC7	CALL	SHOTAD	IX+8 = CD4BC7 (実際のリマ ンタル設定)
	05DB DD350A	DEC	(IX+10)	IX+10 = DD350A (実際のリマ ンタル設定)
	05DE 2002	JR	NZ,RV1	IX+0 = 2002 (実際のリマ ンタル設定)
	05E0	RVEND: ;ReVive END		
	05E0 E1	POP	HL	HL (使用しない)
	05E1 C9	RET		
	05E2	RV1: ;ReVive 1		
	05E2 DD6E04	LD	L,(IX+4)	IX+4 = DD6E04 (実際のリマ ンタル設定)
	05E5 DD6605	LD	H,(IX+5)	IX+5 = DD6605 (実際のリマ ンタル設定)
	05E8 DD750C	LD	(IX+12),L	IX+12 = DD750C (実際のリマ ンタル設定)
	05EB DD740D	LD	(IX+13),H	IX+13 = DD740D (実際のリマ ンタル設定)
	05EE E1	POP	HL	HL (使用しない)
	05EF 23	INC	HL	IX+4, IX+5 には、常に REVIVE が実行され るような常態のコメント・ポインタとなる (SCLP 内の @JUMP となっている)
	05F0 DD7504	LD	(IX+4),L	IX+4 = DD7504 (実際のリマ ンタル設定)
	05F3 DD7405	LD	(IX+5),H	IX+5 = DD7405 (実際のリマ ンタル設定)
19520	05F6 C9	RET		

● APPENDIX

1. MF-ASM2...PC88シリーズ用アセンブラ
2. インストラクション表...いわゆる
3. ツール...Game Programming Kits
4. マシン語命令小辞典...御一読アレツと!



1. MF-ASM2…PC88シリーズ用アセンブラ

『MF-ASM2』は、すでにテープ版の商品としてアスキーから発売されている『MF-ASM』の改良版です。従来のテープ版の『MF-ASM』では、MF-ASM 自身のプログラムをグラフィックのグリーン面の V-RAM に置いていたため、グラフィックを表示すると『MF-ASM2』のプログラムが破壊されてしまいました。ゲームでは、作成したプログラムのテストにグラフィック画面の表示が必要不可欠ですから、このままではテストのたびに『MF-ASM2』をロードしなければなりません。そのため、本書では『MF-ASM2』をゲーム・プログラム作成に使いやすいうように改良し、『MF-ASM2』自身のプログラムは裏 RAM に置くようにしました。テープ版の『MF-ASM』をお持ちの方は、めんどうでもテストのたびに『MF-ASM』をロードし直してください。

1 MF-ASM の起動方法

ここには2つのマシン語プログラムのダンプ・リストと1つの BASIC リストがありますが、ダンプリストの内、1つは『MF-ASM』のプログラム、もう1つは『AUTOQ』で AUTO 命令実行時に自動的に注釈()をつけるためのプログラムです。両方共、モニタの E コマンドで入力するのですが、入力ミスのないようにするため、入力し終えたら BASIC で書かれたチェックサムプログラム CHECK でかならずチェックサムの確認をしてください。入力の手順は、次のようになります。

```

MON  [F1]
h] EB900  [F2]

B900  .....
:
:
:
:
CE20  .....

      < List MF-ASM2 を入力 >

      < [STOP] または [ESC] により h] の状態に戻る >

h] EF2E0  [F3]
F2E0  .....

      < List AUTOQ を入力 >

```


<[STOP] または [ESC] により h] の状態に戻る>

h] ^ b

<List CHECK(チェックサム用プログラム)を入力>

RUN 

start address=&HB900 

end address=&HCE25 

printer (y/n)?プリントアウトするか否かを答え 

B900 : : XXX

⋮

<List MF-ASM2 のチェックサムと比較する>

CE20 : : XXX

チェックサムに間違いがあれば修正し、OK ならプログラムをセーブ

テープの場合 : h] WMFASM2, B900, CE25 

ディスクの場合 : BSAVE "MFASM2", &HB900, &H1526

List AUTOQ も同様にチェックし、OK ならプログラムをセーブ

テープの場合 : h] WAUTOQ, F2E0, F2F4 

ディスクの場合 : BSAVE "AUTOQ", &HF2E0, &H15 

マシン語プログラムの入力完了した後、それぞれの先頭アドレスから実行させることにより、プログラムの転送およびフックの書き換え等を行ないます。プログラムの実行に際しては、G コマンドで走らせるのではなく、かならず BASIC から呼ぶようにしてください。また、実行前には『CLEAR, &HB8FF』とする必要がありますが、本書中のマシン語プログラムでは B500H 番地以降を使うため、このままでは実行後に『CLEAR, &HB4FF』と再宣言しなければなりません。ですから、本書のプログラムに限り、最初から『CLEAR, &HB4FF』とするようにしてください。すなわち、

MFASM の実行 : CLEAR, &HB4FF : DEF USR=&HB900 : A=USR(0) 

AUTOQ の実行 : DEF USR=&HF2E0 : A=USR(0) 

とすればいいのです。これにより、『MF-ASM2』のメイン・プログラムは、裏 RAM(0000_H ~15FF_H 番地…フルに使用されているわけではない)に転送され、『CMD』や『AUTO(

付き)』が使用できるようになります。また、裏 RAM へのプログラム転送に伴い、BASIC の注釈(')文で作られるソース・プログラムは、1800_H番地から作成されるように、フックが書き換えられています。『MF-ASM2』を使用する場合には、プログラムのロード後にこの命令を実行することを忘れないようにしてください。


② MF-ASM2 の利用方法

マシン語プログラムとは、最終的に本体のメモリに記憶された状態になって、初めて実行可能となります。『MF-ASM2』をロード/実行してから、実際に実行可能のプログラムを作成するには、次のような順序を踏んでいくことになります。

- 1 - ソース・プログラムの作成

BASIC の注釈(')文で、Z80 ニーモックによるマシン語プログラムを作ります。使用できる文字は大文字だけ(コメント文は規制なし)に制限されるほか、- 3 -に示されるような文法上のルールがあります。作成したソース・リストのセーブ/ロードは、BASIC プログラムと同じです。

- 2 - アセンブル

『CMD 』とすることにより、まずソース・プログラムからラベル・テーブルがグラフィック・V-RAM のグリーン面に生成され(PASS 1)、次いでオブジェクト・プログラフがブルー面に生成されます(PASS 2)。この時、プログラムにエラーがあれば、エラー番号+(エラーの行番号)+エラー行が表示されます。

エラー番号	エラーの種類
20	文法上の誤り
10	オペラントが不適當
08	存在しない命令
04	ディスプレイメントが不適當(JR の範囲を越えている)
02	ラベルの未定義
01	ラベルの多量定義

もし、この時点でエラーがあれば、ソース・プログラムの見直し/修正をし、エラーがなければ- 3 -に進みます。BASICに戻るには、『Option?』に対し **STOP** を押し、『RETURN TO BASIC MONITOR(B/M)?』には B を入力してください。エラー表

示が出なくなるまで、修正/アセンブルの作業を繰り返します。

- 3 - オプション・コマンド

オプションには、l, p, s, o, e の5つがあり、その内容は次のようになっています。

コマンド	内 容
l	アセンブル・リストを出力
p	プリンタへの出力スイッチ
s	ラベルのソート出力
o	生成されたオブジェクトをメイン・メモリへロード
e	エラー行をソース・プログラムの形式で CRT へ出力

p(プリンタへの出力スイッチ)は、lまたはsと組み合わせて使用することにより、CRTに出力したものと同一ものを、プリンタに出力することができます。リストやラベルの出力中には、CTRL+C(STOP)で中断、CTRL+Sで停止ができます。中断時には、Pでプリンタへの出力、QでCRTのみへの出力と切り換えることが可能ですから、必要な部分だけのリストを取ることができます。

グラフィック・V-RAMのブルー面に生成されたマシン語プログラムは、o コマンドによりメイン・メモリへロードされます。この時、『LOAD OFFSET?』と聞いてきますから、必要に応じてオフセット値（例えば、C000_H番地から作成したプログラムを、D000番地にロードする場合は、オフセット値=1000_Hとなる）を入力します。ただし、ロード・アドレスがCLEAR文の第2パラメータによる制限にかかったり、システムのワークエリア(E600_H番地以降)にかかった場合には、エラーとなりロードできません。

③ 文 法

『MF-ASM2』では、ソース・プログラムの作成をBASICの注釈文(行番号+')として行ないます。したがって、リマーク「;」の付いていない行があった場合には、その行は無視されることとなります。これから説明する文法も、すべてこの注釈文(行番号+')で書かれていることを前提にした上でのものですから、くれぐれも間違いのないようにしてください。

(1) 文 字

使用できる文字は、アスキーコードの20_H以上と、TABコードです。"\$"や"'"は特

殊な意味があります。

(2)ステートメント(文)

文字の集まりのことで、ソース・プログラムの基本構成単位です。1 ステートメントの最大文字数は 80 です。

(3)ステートメントの構造

ステートメントの最初の文字が、"*"または";"であれば、単なる注釈文(コメント文)として扱われます。注釈文以外のステートメントは、次のように構成されますが、かならずしもすべてが必要なわけではありません。

書式 **ラベル** : **命令** **オペランド** : **注釈**

オペレーション コメント

《注意点》

- ・ラベルの後には，コロン(:)を付ける。
- ・命令とオペランドの間は，1個以上のスペースをあける。
- ・2つ以上のオペランドは，カンマ(,)で区切る。
- ・命令の前後やオペランド，数などの区切のスペースは，いくらあっても構わない。

(4) ステートメント各部の説明

ラベル

使用可能文字数は6文字で、文字の先頭は"@"か"?"か英字(A~Z)でなければなりません。ラベルの値は、EQU命令の時はオペランドの値、それ以外の時はそのステートメントの先頭アドレスになります。EQU命令以外の時は、コロンの後を省略することも可能です。なお、次に示すレジスタ名とコンディション・コード名は、ラベルとして使用することができません。

A, B, C, D, E, H, L, AF, BC, DE, HL, IX, IY, SP, I, R, Z, C, NZ, NC, P, M, PE, PO

命令

命令には μ PD780 の命令の他に、アセンブラに情報を与える擬似命令があります。命令の書式はサイログ社・Z80 ニーモニックにはほぼ準拠していますが、いくつかの相違点があります。

- ・相対ジャンプ(JR, DJNZ)命令は、そのオペランドに"e"というディスプレイメント(変位置)を与えることになっていますが、ここではアドレス(ラベルでも可)を記述することとし、ディスプレイメントの計算はアセンブラが行ないます。

(例) JR OE000H
 JR NZ, \$+2
 DJNZ LOOP

- ・IN/OUT 命令は、次のように記述します。

(例) IN A, (n)
 OUT (C), r

- ・EX AF, AF' は EX AF, AF のように"'"をつけずに記述します。

オペランド

命令が必要とする情報(レジスタ名や式)を与えますが、命令によってはオペランドが不要のものもあります。式は、項(定数, \$, ラベル)、または項を演算子(+, -, 特殊演算子, R8)で結合したもの、でなければなりません。

16進定数：0～9 と A～F の 16 個の文字で表わし、数字の後には H を、また A～F で始まる時には前にをつけます。

10進定数：16 進数に変換されます。

文字定数：引用符(')で囲まれた 1 文字または 2 文字のことで、値はそのアスキーコードになります。引用符(')を文字定数として使う時は、2 ヶ 1 組(")で 1 文字とします。コントロール・コード(☒=0DH 等)は、10 進定数または 16 進定数で記述します。なお、DB 命令で文字列として使用する分については複数に分けて記述します。

\$ ：現在アセンブル中のステートメントのアドレスを与えます。

ラベル ：そのラベルに定義された値を与えます。

+ ：項と項との加算を行ないます。

- ：項と項との減算を行ないます。項の前に-がつく時は、その項の 2 の補数にとられると考えます。

.R8 ：.R 8 より左に記述されている値 16 ビットを、8 ビットローテイトします。つまり、上位 8 ビットと下位 8 ビットを入れ換えます。

特殊例 : 文字定数または文字列定数の直後に演算子を続けた場合、引用符' 'の中の最後の文字、およびその前の文字との演算を行ないます。

(例) 'CALL'+80H

オペランドに1バイトの値しか必要としない命令においては、オペランドの値の下位8ビットをデータとします。

(例) LD A, 1234H → LD A, 34H となる。

注釈

セミコロン(;)の後は注釈とみなされます。

擬似命令

アセンブラに対して情報を与える命令で、以下のようなものがあります。

ORG nn : オペランド nn で指定されたアドレスから、マシン語コードを生成していきます。nn には、すでに定義されているラベルや、式で記述することもできます。

(例) ORG 0E000H
ORG START

EQU nn : ラベルの後に記述することにより、ラベルに対して nn の値を与えます。nn には、すでに定義されているラベルや、式で記述することもできます。

(例) BLUE: EQU 5CH

DB n : 1バイト単位でデータの設定ができます。オペランドの値が1バイトを越える場合は、下位8ビットがデータになります。また、オペランドが文字列の場合は、各文字がアスキーコードに変換されてデータとなります。オペランドは、すでに定義されているラベルで記述したり、カンマ(,)で区切って複数指定をすることができますが、文字列は最大30文字までです。


```
(例) DB 5CH, 5DH, RED+1
      DB 'ASCII', 0
      DB 'TEST'+80H
```

DW nn : 2 バイト単位でデータの設定ができますが、上位 8 ビットと下位 8 ビットが入れ換わります。文字列は最初の 2 文字だけが有効となり、それ以降は無視されます。オペランドが 1 文字または 1 バイトの値で記述されている時は、下位 8 ビットが 0 になります。また、オペランドはすでに定義されているラベルで記述したり、カンマ(,)で区切って複数指定することができます。

```
(例) DW 1234H
      DW @JUMP, SKY1
      DW 'AB'
```

DS nn : オペランド nn の値の大きさの領域(内容は不定)を、プログラム中に確保します。オペランドはすでに定義されているラベルで記述したり、カンマ(,)で区切って複数指定することができます。

```
(例) DS 5
```

END : アセンブラに対して、プログラムの終了を指示します。これ以降にあるプログラムは、アセンブルされません。

4 MF-ASM2 のプログラム

MF-ASM2のプログラム		mfasm2
B900	: 00 00 00 CD 98 BA D0 CD CB BA 08 3E 0E D3 53 CD	: 85B
B910	: 04 BB C9 00 00 00 CD CD B9 CD 66 CC 50 41 53 53	: 711
B920	: 2D 31 0D 0A 00 3E FF CD 0F BC 00 00 00 CD 66 CC	: 549
B930	: 50 41 53 53 2D 32 0D 0A 00 AF CD 0F BC 3A 97 BF	: 584
B940	: CB 7F C4 92 BD 3A 97 BF CB 67 C2 30 F3 AF 32 9A	: 97F
B950	: BF 3E 80 32 99 BF CD 74 CC CD 66 CC 4F 50 54 49	: 84F
B960	: 4F 4E 20 69 73 20 22 20 6C 20 70 20 73 20 6F 20	: 439
B970	: 65 20 22 0D 0A 4F 70 74 69 6F 6E 20 3F 00 CD D9	: 53C
B980	: 35 CD 92 5F DA A8 B9 0E 00 23 7E B7 28 08 23 CD	: 6B4
B990	: A9 B9 28 F6 18 B7 79 32 97 BF E6 05 20 8C 79 B7	: 817
B9A0	: 28 88 FE 02 28 84 18 98 C9 CB AF FE 4C 20 03 CB	: 787
B9B0	: C1 C9 FE 50 20 03 CB C9 C9 FE 45 20 03 CB D1 C9	: 923

```

B9C0 : FE 53 20 03 CB F9 C9 FE 4F C0 CB E1 C9 AF 32 97 : 9FB
B9D0 : BF CD BF BB 21 00 C0 11 01 C0 01 FF 3F 36 00 ED : 71B
B9E0 : B0 21 00 D0 22 00 C0 CD C3 BB 21 C0 11 01 C0 : 681
B9F0 : 36 00 01 FF 3F ED B0 C3 FF BB 22 CD BE CD BF BB : 983
BA00 : 22 00 C0 C3 FF BB 3A D2 BE B7 28 04 AF 32 DB BE : 886
BA10 : ED 5B CD BE ED 53 31 BF FD 21 DB 8E 21 DB BE ED : A61
BA20 : 5B CD BE AF 47 4E B1 28 2F 23 3A CF BE B7 28 03 : 6FE
BA30 : EB 18 1F C5 05 E5 2A A0 BF 09 CB 7C CA 92 BA E1 : 971
BA40 : ED 5B A0 BF CD BF BB ED B0 ED 53 A0 BF CD FF BB : BB1
BA50 : E1 C1 09 EB ED 53 CD BE 3A CF BE B7 C2 2E BC CD : A58
BA60 : CE 35 28 28 FE 03 20 05 AF 32 97 BF C9 FE 13 20 : 6AA
BA70 : 1B 3A 97 BF 4F CD 83 35 CB AF FE 50 20 04 CB C9 : 7FF
BA80 : 18 06 FE 51 20 06 CB 89 79 32 97 BF CD 64 CA C3 : 7A6
BA90 : 2E BC F1 F1 F1 CD FF BB C3 49 C7 21 00 00 39 11 : 882
BAA0 : 00 B9 ED 52 D8 CD 66 CC 59 4F 55 20 4D 55 53 54 : 735
BAB0 : 20 45 58 45 43 55 54 45 0D 0A 43 4C 45 41 52 20 : 3D1
BAC0 : 2C 26 48 42 38 46 46 0D 0A 00 C9 2A 18 EB 11 03 : 3C1
BAD0 : 00 B7 ED 52 20 0F 21 01 16 22 58 E6 AF 77 23 77 : 57D
BAE0 : 23 22 18 EB C9 CD 66 CC 50 72 6F 67 72 61 6D 20 : 708
BAF0 : 61 6C 72 65 61 64 79 20 65 78 69 73 74 2E 07 0D : 571
BB00 : 0A 00 37 C9 21 33 F3 22 B7 EE 21 AB CD 11 30 F3 : 6E5
BB10 : 01 7E 00 ED B0 21 E9 CC 11 00 80 01 C2 00 CD 2C : 63F
BB20 : BB ED B0 CD 00 80 3A 9F BF D3 70 C9 DB 70 32 9F : 965
BB30 : BF 3E 80 D3 70 C9 CD 3B C7 CA 67 C7 CD 36 C7 CA : 9E4
BB40 : 67 C7 3A CF BE B7 28 17 CD 79 BB 20 24 01 08 00 : 639
BB50 : EB 21 C7 BE ED B0 1B 1B ED 53 2F BF C3 FF BB CD : 9DC
BB60 : 79 BB CB 7E C2 73 BB 23 CB 7E CD FF BB CB C3 7C : A67
BB70 : C7 CB FE CD FF BB C3 88 C7 CD C3 BB 21 C7 BE 06 : B20
BB80 : 06 11 00 00 78 86 5F 7A CE 00 57 23 10 F6 62 6B : 50C
BB90 : 29 19 7C F6 06 67 29 29 29 29 7E 87 C8 E5 EB : 6BB
BBA0 : 21 C7 BE 06 06 1A E6 7F BE 23 13 20 05 10 F6 E1 : 631
BBB0 : 05 C9 E1 01 08 00 09 7C B7 20 E0 26 C0 18 DC F3 : 6C1
BBC0 : D3 5C C9 F3 D3 5D C9 CD 79 BB 20 25 3A CF BE B7 : 9A8
BBD0 : 28 19 3A D0 BE FE 92 20 10 3A 2E BF B7 28 0A 2A : 603
BBE0 : 2F BF 2B 2B 2B 2B 2B CB FE AF 3D ED 5B CD BE 18 : 765
BBF0 : 0E 23 7E 2F E6 80 28 F3 11 05 00 19 5E 23 56 D3 : 538
BC00 : 5F FB C9 CD C3 BB 2A 2F BF 73 23 72 C3 FF BB 32 : 93D
BC10 : CF BE AF 32 A2 BF 32 D0 BE 21 04 C0 22 A0 BF 21 : 816
BC20 : 00 D0 22 CD BE 22 31 BF 2A 58 E6 22 72 BE 2A 72 : 6E5
BC30 : BE 22 70 BE 21 D0 BE 7E FE 94 CA 21 B0 06 10 CD : 858
BC40 : 1E C8 AF 32 2E BF 2A 70 BE CD A4 44 7E 23 B6 CA : 7E2
BC50 : 21 BD CD 9C C7 21 76 BE CD DD C7 CD 7C BD C3 58 : 9F5
BC60 : BA FE 3F DC 67 C7 FE 5B D4 67 C7 CD F1 C7 FE 3A : B19
BC70 : 20 1A 22 2B BF CD 36 BB 3E FF 32 2E BF CD 8A BD : 774
BC80 : CD DD C7 CD 7F BD C3 10 BA CD F1 C7 B7 28 09 FE : A72
BC90 : 20 28 05 FE 09 C4 67 C7 22 2B BF CD 24 C8 21 D0 : 6FC
BCA0 : BE FD 21 DB BE 7E CB 7F 20 32 FD 34 00 23 5E FD : 83E
BCB0 : 73 01 E6 40 C4 64 C1 CD DA C7 CD 7F BD C3 10 BA : 987
BCC0 : CD 74 C7 3A D0 BE FE B0 20 0A FD 34 00 FD 36 01 : 80D
BCD0 : C9 C3 10 BA FE 94 C4 67 C7 C3 10 BA DD 21 D3 BE : 9F6
BCE0 : DD 36 00 00 CD DA C7 CD 7F BD C3 C3 BC 3A D0 BE : 994
BCF0 : E6 1F FE 15 D2 6F BD DD 34 00 CD D6 C5 B7 28 11 : 87F
BD00 : CD 8A B0 DD 34 00 DD 21 D7 BE CD D6 C5 B7 C4 67 : A02
BD10 : C7 DD 21 D3 BE DD 46 01 DD 4E 05 CD 6A BD C3 10 : 871
BD20 : 8A CD BF BB ED 5B 00 C0 2A CD BE E5 B7 ED 52 22 : 9BB
BD30 : 02 C0 CD FF BB E1 4C CD 5B BD 4D CD 5B BD CD 69 : 9C3
BD40 : CC 20 20 20 20 20 20 20 20 20 20 20 20 20 20 : 2AC
BD50 : 20 20 2A 45 4E 44 2A 0D 0A 00 C9 79 CD 7F CC CD : 5A9
BD60 : 0D 3E 79 CD 83 CC CD 0D 3E C9 3A D0 BE E6 1F 87 : 815
BD70 : 21 34 CA 5F 16 00 19 5E 23 56 E8 E9 FE 2A C8 B7 : 6FF
BD80 : C8 FE 3B C8 E3 23 23 23 E3 C9 2A 2B BF 23 22 2B : 745
BD90 : BF C9 DD 21 97 BF 3A 99 BF CB 7F C4 DC CB 21 00 : 944
BDA0 : C0 22 A6 BF CD 60 BE CD CE 35 28 0A FE 03 CA 47 : 846

```

```

B0B0 : BE FE 13 CC 83 35 2A A6 BF CD EC BD CD FF BB D2 : AB1
B0C0 : 47 BE 22 A6 BF 54 5D CD E3 BD D2 00 BE E5 D5 7E : 972
B0D0 : E6 7F 47 1A E6 7F B8 20 04 23 13 18 F2 D1 E1 38 : 731
B0E0 : E6 18 E2 01 08 00 18 03 01 06 00 09 CD C3 BB 7C : 4DB
B0F0 : B7 C8 7E B7 28 ED 23 23 CB 7E 20 EC 2B 2B 37 C9 : 7BA
B000 : CD 09 BE DA A4 BD C3 A7 BD CD 1A BE 3A A3 BF 3D : 974
B010 : 32 A3 BF B7 C0 CD A0 CB 37 C9 CD C3 BB 62 6B 23 : 97E
B020 : 23 CB FE EB ED 5B A4 BF 06 06 7E E6 7F 12 23 13 : 7B9
B030 : 10 F8 13 D5 5E 23 56 E1 CD FF BB CD 3B CC 23 23 : 849
B040 : 23 23 22 A4 BF EB C9 CD FF BB CD A0 CB 21 02 C0 : 921
B050 : 11 08 00 CD C3 BB CB BE 19 7C B7 20 F9 C3 FF BB : 8CF
B060 : CD FF BB CD F0 CA ED 53 A4 BF 3E 05 32 A3 BF C9 : A51
B070 : E6 0F FE 0A DE 69 27 CD 54 4B FE 0D C0 3E 0A CD : 7B7
B080 : 54 4B 3E 0D C9 E3 7E E6 7F CD DC B9 B6 23 E3 F8 : 98F
B090 : 18 F3 21 00 00 1A D6 30 D8 FE 0A 38 08 D6 07 FE : 647
BEA0 : 10 D0 FE 0A D8 29 29 29 29 85 6F 13 18 E7 F5 0F : 66E
BEB0 : 0F 0F 0F CD BC C3 F1 E6 0F FE 0A DE 69 27 77 23 : 76F
BEC0 : C9 13 1A CD B3 C3 1B 1A CD B3 C3 C9 0E 04 37 F5 : 7B8
BED0 : C5 01 0A 10 AF 29 17 B9 38 02 91 2C 10 F7 C1 B7 : 5FE
BEE0 : F5 0D 7C B5 20 EA 18 05 3E 20 CD EA B9 0D F2 ED : 814
BEF0 : C3 F1 D8 C6 30 CD EA B9 18 F7 DD CB FF 46 C8 23 : AD9
BF00 : 7E FE 5E C0 3A F1 CE E6 FC C0 23 EB CD 97 C3 7D : AE7
BF10 : E6 03 32 F1 CE CB 44 C8 CD 8A C3 0D 50 41 55 53 : 811
BF20 : 45 8D C3 83 35 DD CB FF 46 C8 AF 32 EB CE 3A F6 : 9CC
BF30 : CE 87 20 0D DD CB 00 5E 20 07 DD CB 00 46 CA E8 : 77F
BF40 : C4 CD 3B C5 21 A6 D0 CD E9 C9 EB FE 3B 28 50 FE : A41
BF50 : 2A 28 47 AF 32 EB CE FD 21 26 D0 CD 46 C5 21 A6 : 7E6
BF60 : D0 E5 CD B4 C9 D1 FE 3A 20 0B 21 64 D0 CD F7 C4 : A10
BF70 : 06 00 CD 03 C5 21 6B D0 CD F7 C4 06 20 CD 03 C5 : 73A
BF80 : 21 70 D0 CD F7 C4 06 00 CD 03 C5 01 83 D0 E5 B7 : 874
BF90 : ED 42 E1 23 30 02 60 69 18 80 00 56 D0 18 03 21 : 528
BFA0 : 64 D0 CD F7 C4 01 A5 D0 CD 32 C1 CB 7F CA B3 C0 : A79
BFB0 : CB 7D CA 4A C0 CB 70 CA 08 C0 CB 71 CA EB BF CB : A64
BFC0 : 59 20 20 CB 58 20 0A 1E 40 78 CD 59 C1 79 C3 52 : 631
BFD0 : C1 CD 23 C1 1E 47 78 E6 04 28 02 CB DB FD 73 01 : 77A
BFE0 : C3 64 C1 CD 26 C1 1E 57 79 18 EC CB 6D CA 70 C7 : 8C7
BFF0 : CD 48 C1 79 CD 09 C1 28 06 1E 0A 50 C3 1C C0 1E : 649
C000 : 46 78 CD 59 C1 C3 F0 C1 CB 6D C2 70 C7 CB 6C 1E : 99F
C010 : F9 28 28 78 CD 09 C1 28 11 1E 02 51 FE 80 28 02 : 5AA
C020 : CB E3 FD 73 01 7A CD 27 C1 C9 1E 70 CD 45 C1 CD : 945
C030 : 52 C1 CD E4 C0 CD 16 C1 C3 6C C1 78 FE 8B C2 70 : A4B
C040 : C7 CD F3 C0 FD 73 01 C3 6F C1 CB 70 28 16 CB 6D : 95C
C050 : 20 0B 1E 06 CD 48 C1 CD 59 C1 C3 11 C1 CD 26 C1 : 755
C060 : 1E 3A 18 37 CB 6C 28 1F CB 6D C2 70 C7 CD F0 C0 : 7D3
C070 : 1E 36 FE 82 20 06 CD 11 C1 C3 6C C1 DD 7E 06 DD : 7C7
C080 : 77 03 FD 34 00 18 AE 78 CD 05 C1 CB 6D 20 13 1E : 605
C090 : 01 CD 97 C0 C3 6C C1 78 CD 56 C1 FD 34 CD 11 : 880
C0A0 : C1 C9 78 1E 2A E6 C7 FE 82 28 E6 1E 4B CD 97 C0 : 912
C0B0 : C3 64 C1 E6 20 CA 70 C7 7D FE 80 C2 70 C7 79 FE : A5A
C0C0 : C7 1E 32 28 18 CD 05 C1 1E 43 38 08 1E 22 CD DA : 572
C0D0 : C0 C3 6F C1 CD DA C0 C3 64 C1 CD 56 C1 FD 34 00 : A17
C0E0 : CD 16 C1 C9 78 18 01 79 FE 82 20 08 FD 35 00 C9 : 71A
C0F0 : 78 18 01 79 FE 82 C8 FE 92 C8 FE A2 C8 18 2B FE : 953
C100 : 83 20 06 37 C9 FE 8B 28 FA B7 28 1E FE 82 D8 18 : 7C1
C110 : E5 2A D9 BE 18 03 2A D5 BE 22 DD BE 21 DB BE 34 : 829
C120 : 23 73 C9 79 18 01 78 FE C7 C8 F1 C3 70 C7 16 01 : 7F8
C130 : 18 02 16 02 FD 34 00 3A D3 BE 67 E6 03 BA 7C DD : 691
C140 : 6E 04 C8 18 E5 51 18 01 50 7A E6 F8 FE C0 7A C8 : 849
C150 : 18 D8 E6 07 18 08 E6 03 07 E6 07 07 07 07 B3 5F : 501
C160 : 32 DC BE C9 3E ED 18 15 3E CB 18 11 78 18 01 79 : 629
C170 : E6 30 C8 E6 20 20 04 3E DD 18 02 3E FD C5 01 03 : 641
C180 : 00 21 DE BE 11 DF BE ED B8 34 23 77 C1 C9 CD 32 : 867
C190 : C1 CB 6D C2 70 C7 E6 20 1E E3 C2 3B C0 79 FE 82 : 9AF

```



```

C1A0 : 20 07 78 FE 81 1E EB 18 08 FE 83 C2 70 C7 B8 1E : 797
C1B0 : 08 C2 70 C7 FD 73 01 C9 CD 2E C1 CD 5B C2 FE 81 : 960
C1C0 : C2 70 C7 78 CD FF C0 18 4F CD 32 C1 E6 20 C2 70 : 95C
C1D0 : C7 CD 5B C2 78 FE C7 20 20 CB 6D 20 13 CB 5D 20 : 7E1
C1E0 : 0A 7B E6 F8 5F CD 45 C1 C3 52 C1 CB F3 C3 11 C1 : 98E
C1F0 : CD E7 C0 CD 11 C1 C3 6F C1 CB 6D C2 70 C7 CD F0 : AF4
C200 : C0 79 CD 05 C1 7B FE 86 79 20 13 1E 09 E6 C7 FE : 849
C210 : 82 79 20 04 B8 C2 70 C7 CD 56 C1 C3 6C C1 E6 30 : 8BA
C220 : C2 70 C7 78 FE 82 C2 70 C7 7B FE 8E 1E 4A 28 02 : 883
C230 : 1E 42 79 CD 56 C1 C3 64 C1 CD 2E C1 CD 5B C2 28 : 873
C240 : 10 E6 20 C2 32 C0 7B E6 F8 5F CD 48 C1 CD 52 C1 : 938
C250 : C9 CB 6C C2 70 C7 CB F3 C3 16 C1 3A D1 BE 5F 7C : 9F5
C260 : CB 7C C9 CD 2E C1 CD 5B C2 CA 70 C7 E6 20 C2 32 : 9B1
C270 : C0 78 CB 77 20 10 CD 05 C1 7B 1E 08 FE 34 20 02 : 635
C280 : 1E 03 78 C3 18 C2 CD 4A C1 7B EE 30 5F 78 C3 59 : 79A
C290 : C1 CD 2E C1 FE 09 C2 70 C7 CD 5B C2 3A D5 BE FE : A32
C2A0 : 03 D2 70 C7 FE 01 38 06 CB E3 28 02 CB DB FD 73 : 837
C2B0 : 01 C3 64 C1 CD 2E C1 FE 09 C2 70 C7 CD 5B C2 3A : 8C9
C2C0 : D5 BE 57 A3 C2 70 C7 7A C3 5E C1 CD 2E C1 CD 5B : 9C6
C2D0 : C2 CA 70 C7 E6 20 28 16 CD F0 C0 FD 73 01 FE 82 : 975
C2E0 : 28 06 CD 16 C1 CD FB C2 CD 68 C1 C3 6C C1 7B E6 : 9A3
C2F0 : F8 5F CD 48 C1 CD 52 C1 C3 68 C1 23 7E 73 2B 77 : 8AF
C300 : C9 CD 32 C1 FE 0A C2 70 C7 CD 5B C2 3A D5 BE FE : A3F
C310 : 08 D2 70 C7 CD 59 C1 CB 6D 28 16 CD F3 C0 FD 73 : 95E
C320 : 01 FE 82 28 06 CD 11 C1 CD FB C2 CD 68 C1 C3 6F : 900
C330 : C1 7B E6 F8 5F CD 45 C1 18 BB CD 32 C1 FE 82 C2 : A21
C340 : 70 C7 7D FE 28 28 16 FE A0 C2 70 C7 79 FE C1 C2 : 9A9
C350 : 70 C7 1E 40 CD 48 C1 CD 59 C1 C3 64 C1 CD 26 C1 : 8EE
C360 : CD 5B C2 C3 11 C1 CD 32 C1 FE 2A 28 12 FE A2 C2 : 903
C370 : 70 C7 78 FE C1 C2 70 C7 1E 41 CD 45 C1 18 D8 CD : 956
C380 : 23 C1 18 38 FD 34 00 3A D3 BE FE 09 C9 CD 84 C3 : 814
C390 : 28 27 FE A1 28 19 1E C2 FE 12 C2 70 C7 3A D7 FE : 7E7
C3A0 : FE 08 C2 70 C7 78 CD 59 C1 FD 34 00 C3 11 C1 CD : 8F1
C3B0 : F0 C0 FD 36 01 E9 C3 6C C1 FD 34 00 CD 5B C2 C3 : 99B
C3C0 : 16 C1 CD 84 C3 28 2A FE 12 C2 70 C7 3A D7 BE FE : 913
C3D0 : 08 C2 70 C7 78 FE 04 D2 70 C7 F6 04 1E 00 CD 59 : 7C2
C3E0 : C1 32 D1 BE 2A D9 BE 18 0B CD 2E C1 FE 09 C2 70 : 85B
C3F0 : C7 2A D5 BE ED 5B CD BE 13 13 B7 ED 52 CD 5B C2 : 95D
C400 : 38 0D B7 C2 63 C7 7D FE 80 D2 63 C7 C3 19 C1 3C : 8B8
C410 : C2 63 C7 7D FE 80 DA 63 C7 C3 19 C1 CD 84 C3 1E : 9BA
C420 : C4 C2 98 C3 C3 B9 C3 CD 2E C1 FE 11 C2 70 C7 1E : A02
C430 : C0 78 C3 59 C1 CD 87 C3 C2 70 C7 3A 2E BF B7 CA : 9CD
C440 : 78 C7 3A D2 BE E6 01 C0 3A CF BE B7 ED 5B D5 BE : A09
C450 : C2 03 BC F1 C3 14 BA CD B7 C3 C2 70 C7 3A A2 BF : 9AE
C460 : B7 2A D5 BE ED 5B CD BE 20 0F 3D 32 A2 BF E5 21 : 84C
C470 : 00 D0 ED 52 E1 D2 FA B9 B7 ED 52 DA 70 C7 EB 19 : A80
C480 : 22 CD BE 2A A0 BF 19 22 A0 BF C9 C3 70 C7 47 CD : 8A7
C490 : 87 C3 C8 C3 67 C7 2A CD BE 22 31 BF 21 00 00 22 : 70D
C4A0 : 9D BF CD CB C5 CD 8E C4 ED 5B D5 BE 2A 9D BF 19 : A52
C4B0 : 22 9D BF 78 B7 CA C0 C4 CD 19 C8 CD 8A BD 18 E2 : 9B7
C4C0 : 3A D2 BE B7 C2 18 BA EB 2A CD BE 19 DC 5F C7 22 : 8F2
C4D0 : CD BE 2A A0 BF 19 22 A0 BF C3 18 BA 3E 01 32 F7 : 7AB
C4E0 : C4 3C 32 FA C4 CD CB C5 CD 8E C4 FD 34 00 FD 34 : 9CE
C4F0 : 00 ED 5B D5 BE FD 73 01 FD 72 02 78 B7 CA 06 BA : 876
C500 : CD 19 C8 CD 8A BD 3A FA C4 3C 18 D2 3E 01 32 34 : 785
C510 : C5 CD CB C5 CD 8E C4 3A D5 BE CD 2F C5 78 B7 CA : AC8
C520 : 06 BA 3E 97 32 D0 BE CD 19 C8 CD 8A BD 18 E2 FD : 90E
C530 : 34 00 FD 77 01 E5 21 34 C5 34 E1 C9 F1 CD 90 C5 : 899
C540 : 28 06 4F CD 2F C5 18 F5 B7 CC 67 C7 CD 8D BD AF : 8C2
C550 : 32 D0 BE D0 71 02 CD 45 C6 CD 8E C4 3A D5 BE E5 : 9B9
C560 : 21 34 C5 35 CD 32 C5 E1 18 B3 DD CB 00 DE 3A D0 : 84F
C570 : BE FE 97 28 C7 11 00 00 CD 90 C5 28 0D 5F CD 90 : 766
C580 : C5 28 07 53 5F CD 90 C5 20 FB B7 CA 67 C7 18 13 : 7BD
C590 : 23 7E FE 27 28 02 B7 C9 23 7E FE 27 28 F8 2B 7E : 6FF

```

```

C5A0 : FE 27 C9 22 2B BF CD 76 C6 CD 8A BD C3 45 C6 06 : 8EB
C5B0 : 00 FE 2B 28 04 FE 2D C0 05 F1 78 23 32 2D BF DD : 6CC
C5C0 : CB 00 DE CD FF C5 C2 20 C6 18 23 DD 21 D3 BE FD : 9A9
C5D0 : 21 0B BE DD 34 00 AF 32 2D BF CD DA C7 FE 28 20 : 84C
C5E0 : 05 DD CB 00 EE 23 CD FF C5 20 22 CD AF C5 FE 27 : 8F7
C5F0 : CA 6A C5 FE 24 ED 5B CD BE CA A6 C5 C3 67 C7 CD : AE1
C600 : F1 C7 22 2B BF 47 3A C7 BE FE 20 78 C9 3A D0 BE : 8F1
C610 : CB 6F 28 06 CD 23 C7 CA A3 C6 CD 2D C7 CA AF C6 : 952
C620 : CD 2D C7 CA 74 C7 CD 23 C7 CA 74 C7 3A C7 BE FE : A3F
C630 : 3F 30 09 CD 93 C6 B7 CA 67 C7 18 0D CD C7 BB CC : 88D
C640 : 7C C7 CD 76 C6 DD CB 00 DE CD DA C7 FE 2E 20 12 : 99E
C650 : 23 CD F4 C7 CD DD C7 CD 14 C7 DA 67 C7 CD 08 C7 : A68
C660 : 18 E7 FE 29 28 05 CD AF C5 18 38 DD CB 00 6E CA : 7C4
C670 : 67 C7 23 C3 D6 C6 3A 2D BF B7 28 09 AF 32 2D BF : 78B
C680 : 67 6F ED 52 EB DD 66 03 DD 6E 02 19 DD 74 03 DD : 7DD
C690 : 75 02 C9 DD CB 00 DE CD 8E CC B7 C8 EB CD 76 C6 : A60
C6A0 : 3E FF C9 DD CB 00 6E C2 67 C7 2A 2B BF 18 27 2A : 789
C6B0 : 2B BF DD CB 00 FE DD CB 00 6E 28 1A DD 7E 01 FE : 842
C6C0 : C1 28 0A CB 77 C2 67 C7 E6 30 C2 49 C6 CD DD C7 : 97D
C6D0 : FE 29 C2 67 C7 23 DD 7E 00 E6 A0 FE A0 20 1B DD : 8D1
C6E0 : 7E 03 B7 CA F3 C6 3C C2 5F C7 DD CB 02 7E CA 5F : 930
C6F0 : C7 18 07 DD CB 02 7E C2 5F C7 CD DD C7 B7 C8 FE : 9E4
C700 : 2C C8 D6 3B C8 C3 67 C7 00 00 00 DD 66 02 DD 6E : 74E
C710 : 03 C3 8C C6 21 30 CA 11 C7 BE CD 62 C8 D8 7E C8 : 8DE
C720 : 23 18 F4 CD 36 C7 C0 DD CB 00 E6 18 04 CD 3B C7 : 832
C730 : C0 7E DD 77 01 C9 21 1B CA 18 03 21 F3 C9 11 C7 : 732
C740 : BE CD 62 C8 D8 C8 23 18 F5 CD 66 CC 42 55 46 46 : 8A7
C750 : 45 52 20 4F 56 45 52 20 45 52 52 4F 52 00 C9 06 : 46C
C760 : 04 18 06 06 04 18 23 06 20 CD 8A C7 F1 C3 58 BA : 571
C770 : 06 10 18 16 06 10 18 F1 06 20 18 0E 06 02 18 0A : 1D9
C780 : 06 02 18 E5 06 08 18 E1 06 01 2A CD BE 22 31 BF : 4DA
C790 : 3A D2 BE B0 32 D2 BE AF 32 DB BE C9 2A 70 BE CD : 9A4
C7A0 : A4 44 11 72 BE 01 04 00 ED B0 7E FE 8F 20 09 23 : 622
C7B0 : 7E FE 20 28 16 C3 EB C7 FE 3A C2 EB C7 23 7E FE : 99A
C7C0 : 8F C2 EB C7 23 7E FE E9 C2 EB C7 23 01 50 00 7E : 8F1
C7D0 : B7 ED A0 C8 EA CF C7 AF 12 C9 2A 2B BF 22 2B BF : 936
C7E0 : 7E FE 09 28 03 FE 20 C0 23 18 F2 00 00 F1 C3 2E : 69D
C7F0 : BC CD DD C7 E5 06 06 21 C7 BE E5 3E 20 CD 1F C8 : 8BB
C800 : D1 E1 06 06 7E FE 30 D8 FE 3A 38 06 FE 3F D8 FE : 8CB
C810 : 5B D0 12 13 23 10 ED 7E C9 06 04 21 D3 BE AF 77 : 699
C820 : 23 10 FC C9 3A C7 BE FE 59 D2 84 C7 D6 41 DA 84 : 9A0
C830 : C7 87 26 00 6F 11 82 C8 19 5E 23 56 23 4E 23 46 : 508
C840 : EB 11 C8 BE C5 CD 62 C8 C1 DA 84 C7 20 08 5E 23 : 8CD
C850 : 56 ED 53 D0 BE C9 23 23 E5 ED 42 E1 DA 41 C8 C3 : 9CE
C860 : 84 C7 1A 4F 7E B7 28 0F 47 E6 7F B9 23 13 37 3F : 631
C870 : 20 08 CB 78 28 EC 1A D6 20 C9 78 E6 80 C0 7E 23 : 797
C880 : 18 F9 B4 C8 C0 C8 C4 C8 E6 C8 16 C9 F2 C9 F2 C9 : BA4
C890 : 28 C9 2D C9 49 C9 F2 C9 4F C9 F2 C9 64 C9 6C C9 : 9E9
C8A0 : 8B C9 F2 C9 94 C9 D2 C9 F2 C9 F2 C9 F2 C9 F2 C9 : CF3
C8B0 : EE C9 F2 C9 44 C3 83 8E 44 C4 83 86 4E C4 84 A6 : 9D7
C8C0 : 49 D4 89 46 41 4C CC AF CD 43 C6 00 3F D0 84 BE : 81B
C8D0 : 50 C4 40 A9 50 44 D2 40 B9 50 C9 40 A1 50 49 D2 : 7C1
C8E0 : 40 B1 50 CC 00 2F 41 C1 00 27 C2 97 00 C3 97 00 : 618
C8F0 : 45 C3 86 35 45 46 C2 97 00 45 46 CD 97 00 45 46 : 621
C900 : D7 96 00 45 46 D3 95 00 C9 00 F3 4A 4E DA 8E 10 : 72C
C910 : D3 95 00 D7 96 00 C9 00 FB 4E C4 94 00 51 D5 92 : 7F7
C920 : 00 D8 81 E3 58 D8 00 D9 41 4C D4 00 76 CD C7 46 : 7F6
C930 : CE 8A DB 4E C3 85 34 4E C4 40 AA 4E 44 D2 40 BA : 857
C940 : 4E C9 40 A2 4E 49 D2 40 B2 D0 AC C3 D2 AD 18 C4 : 8EE
C950 : 80 00 44 C4 40 A8 44 44 D2 40 B8 44 C9 40 A0 44 : 6F3
C960 : 49 D2 40 B0 45 C7 40 44 4F D0 00 00 D2 84 B6 52 : 718
C970 : C7 93 00 54 44 D2 40 BB 54 49 D2 40 B3 55 D4 8B : 7D5
C980 : D3 55 54 C4 40 AB 55 54 C9 40 A3 4F D0 82 C1 55 : 837
C990 : 53 C8 82 C5 45 D3 89 86 45 D4 B0 C9 45 54 C9 40 : 8BD

```



```

C9A0 : 4D 45 54 CE 40 45 CC 88 16 4C C1 00 17 4C C3 88 : 65E
C9B0 : 06 4C 43 C1 00 07 4C C4 40 6F D2 88 1E 52 C1 00 : 5A7
C9C0 : 1F 52 C3 88 0E 52 43 C1 00 0F 52 C4 40 67 53 D4 : 613
C9D0 : 91 C7 42 C3 83 9E 43 C6 00 37 45 D4 89 C6 4C C1 : 833
C9E0 : 88 26 52 C1 88 2E 52 CC 88 3E 55 C2 84 96 4F D2 : 7AD
C9F0 : 84 AE 00 C1 C7 C2 C0 C4 C2 C5 C3 C8 C4 CC C5 C9 : B30
CA00 : C8 D2 CC C3 C1 42 C3 80 44 C5 81 48 CC 82 53 D0 : 9B2
CA10 : 8B 49 D8 92 49 D9 A2 41 C6 83 00 4E DA 00 DA 01 : 78F
CA20 : 4E C3 02 C3 03 50 CF 04 50 C5 05 D0 06 CD 07 00 : 5C0
CA30 : 52 B8 00 00 A8 BF 8E C1 B8 C1 C9 C1 39 C2 63 C2 : 8E3
CA40 : 63 C2 91 C2 CB C2 01 C3 3A C3 66 C3 8D C3 C2 C3 : 9C4
CA50 : E9 C3 1C C4 27 C4 84 C2 35 C4 57 C4 8B C4 96 C4 : 9AA
CA60 : DC C4 0C C5 D0 21 97 BF CD F0 CA D0 CB 00 46 20 : 95A
CA70 : 1B 3A D2 BE B7 C8 D0 CB 00 56 28 10 D0 CB 00 8E : 7D0
CA80 : EB CD 31 CC 36 27 23 00 EB C3 91 CB D0 CB 02 7E : 867
CA90 : C4 DC CB 21 76 BE CD D0 C7 B7 CA 94 CB FE 2A CA : B03
CAA0 : 94 CB FE 3B 11 42 BF CA 94 CB 3A D2 BE B7 C2 7D : 993
CAB0 : CB 11 DB BE 1A 13 ED 53 9B BF FE 05 32 98 BF 30 : 7F8
CAC0 : 09 CD FD CA CD 16 CB C3 32 CB CD FD CA CD 14 CB : A4B
CAD0 : CD 32 CB 3A 98 BF D6 04 32 98 BF FE 05 F5 38 02 : 7F0
CAE0 : 3E 04 CD FD CA CD 16 CB EB CD A0 CB F1 D8 18 E3 : A6B
CAF0 : 21 33 BF 54 5D 06 64 3E 20 CD 1F C8 C9 F5 3A D0 : 708
CB00 : BE FE 93 28 0A 21 33 BF ED 5B 31 BF CD 3B CC 21 : 7C1
CB10 : 3B BF F1 C9 3E 04 47 04 05 C8 ED 5B 9B BF 1A 13 : 6DA
CB20 : ED 53 9B BF CD 40 CC E5 2A 31 BF 23 22 31 BF E1 : 888
CB30 : 18 E6 EB 2A 2B BF CD F4 C7 FE 3A 2A 2B BF 20 0E : 7FF
CB40 : 11 42 BF 7E 12 FE 3A 23 13 20 F8 CD D0 C7 11 49 : 6F3
CB50 : BF 01 93 BF 7E B7 28 48 FE 20 28 12 FE 09 28 0E : 64C
CB60 : FE 3B 28 14 CD D3 CB 28 37 22 2B BF 18 E6 11 4E : 6A8
CB70 : BF CD D0 C7 FE 3B 20 1C 11 61 BF 18 17 CD FD CA : 899
CB80 : 36 2F 23 CD 40 CC 36 28 23 CD 31 CC 36 29 23 18 : 546
CB90 : A1 21 76 BE 01 93 BF 7E B7 28 05 CD D3 CB 20 F7 : 82D
CBA0 : 3E 0D 12 13 3E 0A 12 13 AF 12 21 33 BF 7E 23 B7 : 409
CBB0 : 28 10 FE 0C 28 03 CD 0D 3E D0 CB 00 4E C4 D4 3E : 651
CBC0 : 18 E8 D0 35 02 C0 11 33 BF 3E 0C 12 CD A7 CB CD : 742
CBD0 : DC CB C9 12 13 23 78 BA C0 79 BB C9 D0 36 02 3E : 7FA
CBE0 : CD F0 CA CD F4 CB EB CD A0 CB 11 33 BF CD A0 CB : B71
CBF0 : CD F0 CA C9 21 0E CC 11 33 BF 01 23 00 ED B0 D0 : 7EC
CC00 : 34 03 D0 5E 03 16 00 21 79 BF CD 35 CC C9 2A 2A : 5CF
CC10 : 20 20 20 46 2D 41 53 53 45 4D 42 4C 45 52 28 : 3E6
CC20 : 31 29 20 28 50 43 2D 38 38 30 31 29 20 20 20 2A : 2E6
CC30 : 2A ED 5B 74 BE CD 4D CC CD 46 CC 7A CD 40 CC 7B : 937
CC40 : 4F CD 7F CC 77 23 79 CD 83 CC 77 23 C9 E5 EB 11 : 8CA
CC50 : 00 10 11 00 00 29 7B 8F 27 5F 7A 8F 27 57 79 8F : 469
CC60 : 27 4F 10 F1 E1 C9 CD 74 CC E3 7E B7 23 E3 C8 CD : 9E1
CC70 : 0D 3E 18 F5 3E 0D CD 0D 3E 3E 0A CD 0D 3E C9 0F : 4F3
CC80 : 0F 0F 0F E6 0F FE 0A 38 02 C6 07 C6 30 C9 11 C7 : 5C8
CC90 : BE 21 CD BE 06 06 2B 05 7E FE 20 28 F9 04 05 FA : 666
CCA0 : E7 CC FE 48 20 09 3E 29 32 C0 CC 0E 07 18 08 3E : 5BA
CCB0 : 09 32 C0 CC 04 0E 00 21 00 00 C5 29 4D 44 29 29 : 3CB
CCC0 : 09 1A FE 30 38 20 FE 3A 38 0F C1 0C 0D 28 18 FE : 540
CCD0 : 47 30 14 FE 41 38 10 91 C5 D6 30 4F 06 00 09 13 : 4DF
CCE0 : C1 10 D7 3E FF C9 F1 AF C9 18 05 18 0F 18 27 C9 : 763
CCF0 : 21 00 B9 11 10 00 01 29 15 ED B0 C9 F3 3A C2 E6 : 675
CD00 : F6 02 D3 31 21 10 00 11 00 B9 01 29 15 ED B0 3A : 50D
CD10 : C2 E6 D3 31 FB C9 CD 96 F3 4C 4F 41 44 20 4F 46 : 89B
CD20 : 46 53 45 54 3F 20 00 CD 92 5F D8 11 00 00 23 7E : 4D9
CD30 : B7 28 20 FE 47 30 DF FE 41 38 06 CB AF D6 37 18 : 76F
CD40 : 08 D6 30 38 D1 FE 0A 30 CD EB 29 29 29 29 EB B3 : 749
CD50 : 5F 18 DB D9 06 5C 1E 5F D9 F3 CD 8A 80 CD 9E 80 : 898
CD60 : FB D0 CD 96 F3 4C 4F 41 44 20 45 52 52 4F 52 3F : 72A
CD70 : 3F 00 C9 D9 48 ED 79 D9 2A 00 C0 19 EB 21 04 C0 : 73B
CD80 : ED 4B 02 C0 D3 5F C9 78 B1 C8 CD B8 80 7E CD BC : 9F2
CD90 : 80 08 7A FE E6 30 07 08 12 23 13 0B 18 E9 08 37 : 4B8

```



```

CDA0 : C9 D9 48 18 02 D9 4B ED 79 D9 C9 18 4E 00 3E 0E : 6E2
CDB0 : D3 53 22 AC F3 CD 78 F3 CD 13 B9 2A AC F3 CD 96 : 9E4
CDC0 : F3 52 45 54 55 52 4E 20 54 4F 20 42 41 53 49 43 : 518
CDD0 : 20 4F 52 20 4D 4F 4E 49 54 4F 52 20 28 42 2F 4D : 40F
CDE0 : 29 20 3F 00 CD 83 35 CB AF FE 42 C8 FE 4D CA 26 : 7CA
CDF0 : E8 18 C8 CD 8C F3 CD 02 80 18 06 CD 8C F3 CD 04 : 89E
CE00 : 80 3A AB F3 D3 70 C9 DB 70 32 AB F3 3E 80 D3 70 : 980
CE10 : C9 3E 0D CD 0D 3E 3E 0A CD 0D 3E E3 7E B7 23 E3 : 6AA
CE20 : C8 CD 0D 3E 18 F5 00 00 00 00 00 00 00 00 00 : 2ED

```

5 オート・リマーク・プログラム

オート・リマーク・プログラム

autoq

```

F2E0 : 21 EC F2 22 19 ED 3E C3 32 18 ED C9 3A 00 EB 87 : 804
F2F0 : C8 3E 27 DF C9 00 00 00 00 00 00 00 00 00 00 : 2D5

```

6 チェック・サム・プログラム

チェック・サム・プログラム

```

100 '***** CHECKSUM *****
110 DEFINT A-Z
120 DEF FNH$(X,N)=RIGHT$("0000"+HEX$(X),N)
130
140 INPUT "start address = &H",ST$
150 ST=VAL("&H"+ST$): ADR=ST
160 INPUT "end address = &H",EN$
170 EN=VAL("&H"+EN$)
180 IF ST<EN AND ST*EN>=0 THEN 220
190 BEEP: PRINT "チキマセン!!"
200 IF ST*EN<0 THEN PRINT "&H0000 マタハ &H8000 ヲ マタイシ" + "マ" * 3
210 GOTO 140
220 INPUT "printer (y/n)? ",YN$
230 IF YN$="" THEN YN$="n"
240 IF INSTR("Yy",YN$) THEN DEV$="lpt:" ELSE DEV$="scrn:"
250 OPEN DEV$ FOR OUTPUT AS #1
260
270 WHILE EN>ADR
280 FOR J=0 TO 59
290 CSUM=0
300 PRINT #1,FNH$(ADR,4)" : ";
310 FOR I=0 TO 15
320 V=PEEK(ADR+I)
330 CSUM=CSUM+V
340 PRINT #1,FNH$(V,2)" ";
350 NEXT I
360 PRINT #1," : FNH$(CSUM,3)
370 ADR=ADR+16: IF EN<ADR THEN J=59
380 NEXT J
390 IF DEV$="lpt:" THEN PRINT #1,CHR$(12)
400 WEND
410
420 CLOSE

```

2. インストラクション表…いわゆる

	ニーモニック	オペレーション	フ ラ グ						サイクル	アドレス	OPコード			
			S	Z	H	P	V	N			C	76	543	210
8ビット命令	LD r, s	r ← s	1	4	01	.	s	5
	LD r, n	r ← n	2	7	00	r	110	5
			n	.	
	LD r, (HL)	r ← (HL)	1	7	01	.	110	5
	LD r, (IX+d)	r ← (IX+d)	3	19	11	011	101	
					01	r	110	5
			d	.	
	LD r, (IY+d)	r ← (IY+d)	3	19	11	111	101	
					01	r	110	5
			d	.	
	LD (HL), r	(HL) ← r	1	7	01	110	.	5
	LD (IX+d), r	(IX+d) ← r	3	19	11	011	101	
					01	110	.	5
			
	LD (IY+d), r	(IY+d) ← r	3	19	11	111	101	
					01	110	.	5
			d	.	
	LD (HL), n	(HL) ← n	2	10	00	110	110	
			
	LD (IX+d), n	(IX+d) ← n	4	19	11	011	101	
					00	110	110	
			d	.	
			n	.	
	LD (IY+d), n	(IY+d) ← n	4	19	11	111	101	
					00	110	110	
			d	.	
			n	.	
	LD A, (BC)	A ← (BC)	1	7	00	001	010	
	LD A, (DE)	A ← (DE)	1	7	00	011	010	
	LD A, (nn)	A ← (nn)	3	13	00	111	010	
			n	.	
			n	.	
LD (BC), A	(BC) ← A	1	7	00	000	010		
LD (DE), A	(DE) ← A	1	7	00	010	010		
LD (nn), A	(nn) ← A	3	13	00	110	010		
		n	.		
			
LD A, I	A ← I	.	.	.	0	IFF	0	.	2	9	11	101	101	
		.	.	.	0	IFF	0	.			01	010	111	
LD A, R	A ← R	.	.	.	0	IFF	0	.	2	9	11	101	101	
		.	.	.	0	IFF	0	.			01	011	111	
LD I, A	I ← A	2	9	11	101	101	
				01	000	111		
LD R, A	R ← A	2	9	11	101	101	
				01	001	111		
16ビット命令	LD dd, nn	dd ← nn	3	10	00	dd0	001	
			n	.	
			n	.	

	ニーモニック	オペレーション	フラグ						バイト	ステータス	OPコード			
			S	Z	H	P/V	N	C			76	543	210	
16ビット命令	LD IX, nn	IX ← nn	4	14	11 011 101	00 100 001	.	1
			n	.	
	LD IY, nn	IY ← nn	4	14	11 111 101	00 100 001	.	
			n	.	
	LD HL, (nn)	H ← (nn+1) L ← (nn)	3	16	00 101 010	.	n	.
			n	.	
	LD dd, (nn)	dd _H ← (nn+1) dd _L ← (nn)	4	20	11 101 101	01 dd1 011	.	
			n	.	
	LD IX, (nn)	IX _H ← (nn+1) IX _L ← (nn)	4	20	11 011 101	00 101 010	.	[1]
			n	.	
	LD IY, (nn)	IY _H ← (nn+1) IY _L ← (nn)	4	20	11 111 101	00 101 010	.	
			n	.	
	LD (nn), HL	(nn+1) ← H (nn) ← L	2	16	00 100 010	.	n	.
			n	.	
	LD (nn), dd	(nn+1) ← dd _H (nn) ← dd _L	4	20	11 101 101	01 dd0 011	.	
			n	.	
	LD (nn), IX	(nn+1) ← IX _H (nn) ← IX _L	2	20	11 011 101	00 100 010	.	[1]
			n	.	
	LD (nn), IY	(nn+1) ← IY _H (nn) ← IY _L	4	20	11 111 101	00 100 010	.	
			n	.	
8ビット命令	LD SP, HL	SP ← HL	1	6	11 111 001			
	LD SP, IX	SP ← IX	2	10	11 011 101			
					11 111 001			
	LD SP, IY	SP ← IY	2	10	11 111 101			
					11 111 001			
	PUSH qq	(SP-2) ← qq _H (SP-1) ← qq _L	1	11	11 qq0 101			
								
	PUSH IX	(SP-2) ← IX _H (SP-1) ← IX _L	2	15	11 011 101			
					11 100 101			[2]
	PUSH IY	(SP-2) ← IY _H (SP-1) ← IY _L	2	15	11 111 101			
					11 100 101			
	POP qq	qq _H ← (SP+1) qq _L ← (SP)	1	10	11 qq0 001			[2]
4ビット命令	POP IX	IX _H ← (SP+1) IX _L ← (SP)	2	14	11 011 101			
					11 100 001			
	POP IY	IY _H ← (SP+1) IY _L ← (SP)	2	14	11 111 101			
					11 100 001			

	ニーモニック	オペレーション	フ ラ グ							バイト	ステート	OPコード			
			S	Z	H	P	V	N	C			76	543	210	
エクステンジ命令	EX DE, HL	DE ← HL	1	4	11	101	011	
	EX AF, AF'	AF ← AF'	1	4	00	001	000	
	EXX	BC ← BC' DE ← DE' HL ← HL'	1	4	11	011	001	
	EX (SP), HL	H ← (SP+1) L ← (SP)	1	19	11	100	011	
	EX (SP), IX	IX _H ← (SP+1) IX _L ← (SP)	2	23	11	011	101	
	EX (SP), IY	IY _H ← (SP+1) IY _L ← (SP)	2	23	11	111	101	
					11	100	011	
ブロック転送命令	LDI	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1	.	.	0	.	.	0	.	2	16	11	101	101	
												10	100	000	
	LDIR	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1 until BC=0	.	.	0	0	0	0	.	2	21 if BC ≠ 0 16 if BC = 0	11	101	101	
												10	110	000	
	LDD	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1	.	.	0	.	.	0	.	2	16	11	101	101	
												10	101	000	
	LDDR	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1 until BC=0	.	.	0	0	0	0	.	2	21 if BC ≠ 0 16 if BC = 0	11	101	101	
												10	111	000	
ブロックサーチ命令	CPI	A ← (HL) HL ← HL+1 BC ← BC-1	1	.	2	16	11	101	101	
												10	100	001	
	CPIR	A ← (HL) HL ← HL+1 BC ← BC-1 until A = (HL) or BC=0	1	.	2	21 if BC ≠ 0 and A ≠ (HL) 16 if BC = 0 or A = (HL)	11	101	101	
												10	110	001	
	CPD	A ← (HL) HL ← HL-1 BC ← BC-1	1	.	2	16	11	101	101	
8ビット演算命令												10	101	001	
	CPDR	A ← (HL) HL ← HL-1 BC ← BC-1 until A = (HL) or BC=0	1	.	2	21 if BC ≠ 0 and A ≠ (HL) 16 if BC = 0 or A = (HL)	11	101	101	
												10	111	001	
	ADD A, r	A ← A+r	V	0	.	1	4	10	000	r	5
	ADD A, n	A ← A+n	V	0	.	2	7	11	000	n	→
8ビット演算命令	ADD A, (HL)	A ← A+(HL)	V	.	.	1	7	10	000	n	→
	ADD A, (IX+d)	A ← A+(IX+d)	V	0	.	3	19	11	011	101	
												10	000	n	→
												.	d	.	

	ニーモニック	オペレーション	フ ラ グ						バイト	ステート	OPコード			
			S	Z	H	P/V	N	C			76	543	210	
8ビット算術論理演算命令	ADD A, (IY+d)	$A \leftarrow A + (IY + d)$	V	0	1	3	19	11	111	101	
	ADC A, S	$A \leftarrow A + S + CY$	V	0	1			10	000 ^{*1}	110	
	SUB S	$A \leftarrow A - S$	V	1	...				d		
	SBC A, S	$A \leftarrow A - S - CY$	V	1	...				001 ^{*2}		5
	AND S	$A \leftarrow A \cdot S$	P	0	0				010 ^{*2}		5
	OR S	$A \leftarrow A \vee S$	P	0	0				011 ^{*2}		
	XOR S	$A \leftarrow A \oplus S$	P	0	0				100 ^{*2}		
	CP S	$A - S$	V	1	...				110 ^{*2}		
	INC r	$r \leftarrow r + 1$	V	0	...	1	4	00	d	100 ^{*3}	5
	INC (HL)	$(HL) \leftarrow (HL) + 1$	V	0	...	1	11	00	110 ^{*3}	100	
	INC (IX+d)	$(IX + d) \leftarrow (IX + d) + 1$	V	0	...	3	23	11	011	101	
											00	110 ^{*3}	100	
	INC (IY+d)	$(IY + d) \leftarrow (IY + d) + 1$	V	0	...	3	23	11	111	101	
16ビット算術演算命令	DEC S	$S \leftarrow S - 1$	V	1	...				d	101 ^{*4}	
	ADD HL, ss	$HL \leftarrow HL + ss$	0	1	11	00	ss1	001	1
	ADC HL, ss	$HL \leftarrow HL + ss + CY$	V	0	...	1	15	11	101	101	
	SBC HL, ss	$HL \leftarrow HL - ss - CY$	V	1	...	2	15	01	ss1	010	1
	ADD IX, pp	$IX \leftarrow IX + pp$	0	2	15	11	011	101	1
	ADD IY, rr	$IY \leftarrow IY + rr$	0	2	15	00	pp1	001	3
	INC ss	$ss \leftarrow ss + 1$	1	6	11	101	101	1
	INC IX	$IX \leftarrow IX + 1$	2	10	00	ss0	010	
	INC IY	$IY \leftarrow IY + 1$	2	10	11	011	101	
	DEC ss	$ss \leftarrow ss - 1$	1	6	00	100	011	
	DEC IX	$IX \leftarrow IX - 1$	2	10	00	ss1	011	1
	DEC IY	$IY \leftarrow IY - 1$	2	10	11	011	101	
											00	101	011	
アキュムレータ操作命令	DAA	Decimal adjust Acc	P	1	4	00	100	111	
	CPL	$A \leftarrow \bar{A}$	1	1	4	00	101	111	
	NEG	$A \leftarrow \bar{A} + 1$	V	1	...	2	8	11	101	101	
	CCF	$CY \leftarrow \bar{CY}$	0	1	4	01	000	100	
	SCF	$CY \leftarrow 1$	0	1	4	00	111	111	
CPUコントロール命令	NOP	No operation	1	4	00	110	110	
	HALT	CPU halted	1	4	01	110	110	
	DI	IFF = 0	1	4	11	110	011	
	EI	IFF = 1	1	4	11	111	011	

S=r, n, (HL), (IX+d), (IY+d)

*1, 2のところに*2のそれぞれのコードが入り、ADD命令と同様に各命令を展開

*3のところに*4が入り、INC命令と同様に命令を展開

	ニーモニック	オペレーション	フ ラ グ						バイト	ワード	OPコード		
			S	Z	H	P/V	N	C			76	543	210
CPU コントロール命令	IM 0	Set interrupt mode 0	2	8	11	101	101
	IM 1	Set interrupt mode 1	2	8	01	000	110
	IM 2	Set interrupt mode 2	2	8	01	010	110
ロテート・シフト命令	RLCA	$\overline{CY} \rightarrow A$ A	.	.	0	.	0	.	1	4	00	000	111
	RLA	$\overline{CY} \rightarrow A$ A	.	.	0	.	0	.	1	4	00	010	111
	RRCA	$\overline{A} \rightarrow \overline{CY}$ A	.	.	0	.	0	.	1	4	00	001	111
	RRA	$\overline{A} \rightarrow \overline{CY}$ A	.	.	0	.	0	.	1	4	00	011	111
	RLC r		.	.	0	P	0	.	2	■	11	001	011
	RLC(HL)		.	.	0	P	0	.	2	15	00	000 ^{*5}	110 ^{*5}
	RLC(IX+d)	$\overline{CY} \leftarrow \overline{A}$ s = r, (HL), (IX+d), (IY+d)	.	.	0	P	0	.	4	23	11	011	101
	RLC(IY+d)		.	.	0	P	0	.	4	23	11	001	011
	RL s	$\overline{CY} \leftarrow \overline{A}$ s = r, (HL), (IX+d), (IY+d)	.	.	0	P	0	.			00	000 ^{*5}	110 ^{*5}
	RRC s	$\overline{A} \rightarrow \overline{CY}$ s = r, (HL), (IX+d), (IY+d)	.	.	0	P	0	.				001	
	RR s	$\overline{A} \rightarrow \overline{CY}$ s = r, (HL), (IX+d), (IY+d)	.	.	0	P	0	.				011	
	SLA s		.	.	0	P	0	.				100	
	SRA s	$\overline{CY} \leftarrow \overline{A}$ s = r, (HL), (IX+d), (IY+d)	.	.	0	P	0	.				101 ^{*6}	
	SRL s	$\overline{CY} \leftarrow \overline{A}$ s = r, (HL), (IX+d), (IY+d)	.	.	0	P	0	.				111 ^{*6}	
	RLD		.	.	0	P	0	.	2	18	11	101	101
	RRD		.	.	0	P	0	.	2	18	01	101	111
			.	.	0	P	0	.			11	101	101
			.	.	0	P	0	.			01	100	111

	ニーモニック	オペレーション	フラグ						バイト	ステップ	OPコード			
			S	Z	H	P/V	N	C			76	543	210	
ビット操作命令	BIT b, r	$Z \leftarrow r$.	.	1	.	0	.	2	8	11	001	011	
											01	b	r	[5, 6]
	BIT b, (HL)	$Z \leftarrow (HL)_b$.	.	1	.	0	.	2	12	11	001	011	
											01	b	110	[6]
	BIT b, (IX+d)	$Z \leftarrow (IX+d)_b$.	.	1	.	0	.	4	20	11	011	101	
											11	001	011	
											.	d	.	
											01	b	110	6
	BIT b, (IY+d)	$Z \leftarrow (IY+d)_b$.	.	1	.	0	.	4	20	11	111	101	
											11	001	011	
											.	d	.	
ジャンプ・コール命令	SET b, r	$r_b \leftarrow 1$	2	8	11	001	011	
											[11]	b	r	[5, 6]
	SET b, (HL)	$(HL)_b \leftarrow 1$	2	15	11	001	011	
											[11]	b	110	[6]
	SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	4	23	11	011	101	
											11	001	011	
											.	d	.	
											[11]	b	110	[6]
	SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	4	23	11	111	101	
											11	001	011	
											.	d	.	
ジャンプ・コール命令	RES b, s	$s_b \leftarrow 0$ s ≡ r, (HL), (IX+d), (IY+d)	4	23	[11]	b	110	[6]
											[10]	.	.	
	JP nn	$PC \leftarrow nn$	3	10	11	000	011	
											.	n	.	
	JP cc, nn	If cc is true $PC \leftarrow nn$ Otherwise continue	3	10	11	cc	010	7
											.	n	.	
											.	n	.	
	JR e	$PC \leftarrow PC + e$	3	12	00	011	000	
											.	e-2	.	
	JR C, e	If C=0 continue If C=1 $PC \leftarrow PC + e$	3	7 if C=0 12 if C=1	00	111	000	
ジャンプ・コール命令											.	e-2	.	
	JR NC, e	If C=1 continue If C=0 $PC \leftarrow PC + e$	3	7 if C=1 12 if C=0	00	110	000	
											.	e-2	.	
	JR Z, e	If Z=0 continue If Z=1 $PC \leftarrow PC + e$	3	7 if Z=0 12 if Z=1	00	101	000	
											.	e-2	.	
	JR NZ, e	If Z=1 continue If Z=0 $PC \leftarrow PC + e$	3	7 if Z=1 12 if Z=0	00	100	000	
											.	e-2	.	
	JP (HL)	$PC \leftarrow HL$	1	4	11	101	001	
	JP (IX)	$PC \leftarrow IX$	2	8	11	011	101	
											11	101	001	
ジャンプ・コール命令	JP (IY)	$PC \leftarrow IY$	2	8	11	111	101	
											11	101	001	

*5のところに*6のそれぞれのコードが入り、RLC命令と同様に各命令を展開

*7のところに*8のそれぞれのコードが入り、SET命令と同様に各命令を展開

	ニーモニック	オペレーション	フ ラ グ							バイト	ステート	OPコード			
			S	Z	H	P/V	N	C	76			543	210		
ジャンプ・コール命令	DJNZ e	B ← B - 1 if B = 0 continue if B ≠ 0 PC ← PC + e	2	8 if B = 0 13 if B ≠ 0	00 010 000 ← e - 2			
	CALL nn	(SP - 1) ← PC _H (SP - 2) ← PC _L PC ← nn	3	17	11 001 101 ← n			
	CALL cc, nn	If cc is false continue otherwise same as CALL nn	3	10 if cc is false 17 if cc is true	11 cc 100 ← n			[7]
	RET	PC _L ← (SP) PC _H ← (SP + 1)	1	10	11 001 001			
	RET cc	If cc is false continue otherwise same as RET	1	5 if cc is false 11 if cc is true	11 cc 000			[7]
	RETI	Return from interrupt	2	14	11 101 101 01 001 101			
	RETN	Return from non maskable interrupt	2	14	11 101 101 01 000 101			
	RST p	(SP - 1) ← PC _H (SP - 2) ← PC _L PC _H ← 0 PC _L ← P	1	11	11 1 111			[8]
	入出力命令	IN A, n	A ← (n) A ₀₋₇ ← n A ₈₋₁₅ ← A	2	11	11 011 011 ← n		
IN r, (C)		r ← (C) if r = 110 only the flags will be affected A ₀₋₇ ← C A ₈₋₁₅ ← B	P	0	.	2	12	11 101 101 01 r 000			[5]
INI		(HL) ← (C) B ← B - 1 HL ← HL + 1 A ₀₋₇ ← C A ₈₋₁₅ ← B	1	.	2		11 101 101 10 100 010			
INIR		(HL) ← C B ← B - 1 HL ← HL - 1 until B = 0 A ₀₋₇ ← C A ₈₋₁₅ ← B	.	1	.	.	.	1	.	2	21 if B ≠ 0 16 if B = 0	11 101 101 10 110 010			
IND		(HL) ← (C) B ← B - 1 HL ← HL - 1 A ₀₋₇ ← C A ₈₋₁₅ ← B	1	.	2	16	11 101 101 10 101 010			
INDR		(HL) ← (C) B ← B - 1 HL ← HL - 1 until B = 0 A ₀₋₇ ← C A ₈₋₁₅ ← B	1	1	.	2	21 if B ≠ 0 16 if B = 0	11 101 101 10 111 010			
OUT n, A		n ← A A ₀₋₇ ← n A ₈₋₁₅ ← A	2	11	11 010 011 ← n			

	ニーモニック	オペレーション	フ ラ グ						バイト	ステート	OPコード			
			S	Z	H	P/V	N	C			76	543	210	
入出力命令	OUT (C), r	(C) ← r A ₁₅₋₇ ← C A ₆₋₀ ← B	2	12	11	101	101	
											01	r	001	5
	OUTI	(C) ← (HL) B ← B-1 HL ← HL+1 A ₆₋₇ ← C A ₆₋₀ ← B	1	.	2	16	11	101	101	
											10	100	011	
	OTIR	(C) ← (HL) B ← B-1 HL ← HL+1 until B=0 A ₆₋₇ ← C A ₆₋₁₅ ← B	.	1	.	.	1	.	2	21 if B≠0 16 if B=0	11	101	101	
											10	110	011	
	OUTD	(C) ← (HL) B ← B-1 HL ← HL-1 A ₆₋₇ ← C A ₆₋₁₅ ← B	1	.	2	16	11	101	101	
											10	101	011	
	OTDR	(C) ← (HL) B ← B-1 HL ← HL-1 until B=0 A ₆₋₇ ← C A ₆₋₁₅ ← B	.	1	.	.	1	.	2	21 if B≠0 16 if B=0	11	101	101	
											10	111	011	

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
Reg ss	Reg qq	Reg pp	Reg rr	Reg r,s	Bit	cc Condition	Flag P t
B C 00	B C 00	B C 00	B C 00	0 000	0 000	000 N Z Non Zero	Z 00H 000
D E 01	D E 01	D E 01	D E 01	C 001	1 001	001 Z Zero	Z 08H 001
H L 10	H L 10	I X 10	I Y 10	D 010	2 010	010 N C Non Carry	C 10H 010
S P 11	A F 11	S P 11	S P 11	E 011	3 011	011 C Carry	C 18H 011
				H 100	4 100	100 P O Parity Odd	P/V 20H 100
				L 101	5 101	101 P E Parity Even	P/V 28H 101
				A 111	6 110	110 P Sign Positive	S 30H 110
					7 111	111 M Sign Negative	S 38H 111

フ ラ グ	
.	影響受けない
0	リセット
1	セット
x	不定
.	演算結果に従った影響を受ける
P	"1" 偶数パリティ, "0" 奇数パリティ
V	"1" オーバフロー有り, "0" オーバフロー無し
IFF	P/Vフラグ←(IFF)

A	BC-1=0 ならば P/V=0, その他 P/V=1
B	A=(HL) ならば Z=1, その他 Z=0
C	B-1=0 ならば Z=1, その他 Z=0

d, n	8ビット・イミディエイト・データ
e	相対アドレッシングの変位値
e-2	eの実効変位置

3. ツール…Game Programming Kits

1 パターン・エディタ

パターン・エディタ

pated

```
1000 ***** PATTERN EDITOR ***** by T.Hidaka on 1985.5.31
1010 CLEAR ,&HB4FF
1020 SCREEN 0,3:WIDTH 80,25:CONSOLE 0,25,0,1:DEFINT A-Z
1030
1040 FOR N=0 TO 198
1050   READ A$:POKE &HBF00+N,VAL("&H"+A$)
1060 NEXT
1070
1080 CLS:PRINT "パターン サイズ" (Max.X=56,Max.Y=24)
1090 PRINT:INPUT "X(DOT),Y(DOT) ";MX,MY
1100 IF MX<1 OR MX>56 THEN 1070
1110 IF MY<1 OR MY>24 THEN 1070
1120 POKE &HBF01,(MX-1)*8+1:POKE &HBF00,MY:CLS
1130 DEF USR=&H4290:DIM CLR%(117)
1140 FOR N=1 TO 8
1150   LINE(N*16,193)-(N*16+6,199),N-1,BF
1160 NEXT
1170 LINE(144,193)-(150,199),7,B
1180 FOR N=1 TO 9
1190   GET@(N*16,193)-(N*16+6,199),CLR%(13*(N-1))
1200 NEXT
1210 SCREEN 0,0:LOCATE 2,23:PRINT "0 1 2 3 4 5 6 7 8"
1220 LOCATE 16,22:PRINT "◆"
1230 LOCATE 23,0
1240 FOR N=1 TO MX : PRINT HEX$(N MOD 10); : NEXT
1250 FOR N=1 TO MY : LOCATE 21,N:PRINT USING "##";N; : NEXT
1260 LINE(182,7)-(184+MX*8,MY*8+7),7,B
1270 LOCATE 0,6
1280 PRINT "5: SET"
1290 PRINT "0: RESET":PRINT
1300 PRINT "E: End"
1310 PRINT "C: Color"
1320 PRINT "A: Auto paint"
1330 PRINT "X: X ハンテン"
1340 PRINT "Y: Y ハンテン"
1350 PRINT "S: Shift"
1360 PRINT "P: Pallet"
1370 PRINT "L: Large"
1380 CX=23:CY=1:CC=7:CP=7:CONSOLE 19,3
1390
1400
1410
1420 *LOOP
1430 LOCATE CX,CY:A$=INPUT$(1)
1440 ON INSTR("2468137950",A$) GOSUB *D2,*D4,*D6,*D8,*D1,*D3,*D7,*D9,*ST,*RS
1450 IF A$="E" OR A$="e" THEN *MAKE.DATA
1460 IF A$="C" OR A$="c" THEN *CHANGE.CC
1470 IF A$="A" OR A$="a" THEN *A.PAINT
1480 IF A$="X" OR A$="x" THEN *X.HANTEN
1490 IF A$="Y" OR A$="y" THEN *Y.HANTEN
1500 IF A$="S" OR A$="s" THEN *SHIFT
1510 IF A$="P" OR A$="p" THEN *PALLET
```

```

1520 IF A$="L" OR A$="I" THEN *LARGE
1530 IF (INP(8) OR 191)=191 THEN GOSUB *ST
1540 GOTO *LOOP
1550 '
1560 '
1570 *D2
1580 IF CY<MY THEN CY=CY+1
1590 RETURN
1600 *D4
1610 IF CX>23 THEN CX=CX-1
1620 RETURN
1630 *D6
1640 IF CX<MX+22 THEN CX=CX+1
1650 RETURN
1660 *D8
1670 IF CY>1 THEN CY=CY-1
1680 RETURN
1690 *D1
1700 IF CX<>23 AND CY<>MY THEN CX=CX-1:CY=CY+1
1710 RETURN
1720 *D3
1730 IF CX<>MX+22 AND CY<>MY THEN CX=CX+1:CY=CY+1
1740 RETURN
1750 *D7
1760 IF CX<>23 AND CY<>1 THEN CX=CX-1:CY=CY-1
1770 RETURN
1780 *D9
1790 IF CX<>MX+22 AND CY<>1 THEN CX=CX+1:CY=CY-1
1800 RETURN
1810 *ST
1820 PUT@(CX*8,CY*8),CLR%(CC*13),PSET:PSET(CX-23,CY-1),CP
1830 RETURN
1840 *RS
1850 PUT@(CX*8,CY*8),CLR%(0),PSET:PSET(CX-23,CY-1),0
1860 RETURN
1870 '
1880 *CHANGE.CC
1890 LOCATE 0,19:PRINT "COLOR NO.? ";
1900 A=USR(0):A$=INKEY$
1910 IF A$<"0" OR A$>"8" THEN 1900
1920 GOSUB *MSGCLS:LOCATE 2,22:PRINT SPC(17);
1930 CC=VAL(A$):LOCATE CC*2+2,22:PRINT "◆";
1940 IF CC=8 THEN CP=0 ELSE CP=CC
1950 GOTO *LOOP
1960 '
1970 '
1980 *A.PAINT
1990 LOCATE 0,19:PRINT "ALL(A) OR PART(P) ? ";
2000 A=USR(0):A$=INKEY$:IF A$="" THEN 2000
2010 IF A$="A" OR A$="a" THEN *ALL.PAINT
2020 IF A$="P" OR A$="p" THEN *PART.PAINT
2030 GOSUB *MSGCLS
2040 GOTO *LOOP
2050 '
2060 *ALL.PAINT
2070 GOSUB *MSGCLS
2080 FOR N=1 TO MY
2090   FOR M=1 TO MX
2100     A=USR(0):LOCATE 22+M,N
2110     PUT@(176+M*8,N*8),CLR%(CC*13),PSET:PSET(M-1,N-1),CP
2120   NEXT
2130 NEXT
2140 GOTO *LOOP

```

```

2150
2160 *PART.PAINT
2170 GOSUB *MSGCLS
2180 B$="8":GOSUB *FROMTO
2190 IF C1=9 THEN 2300
2200 C2C=C2:IF C2<0 OR C2>8 THEN 2300 ELSE IF C2=8 THEN C2C=0
2210 FOR N=1 TO MY
2220   FOR M=1 TO MX
2230     A=USR(0):LOCATE 22+M,N
2240     C3=POINT(180+M*8,N*8)
2250     IF C3=7 AND POINT(180+M*8,N*8+1)=0 THEN C3=8
2260     IF C3<>C1 THEN 2280
2270     PUT@(176+M*8,N*8),CLR%(C2*13),PSET:PSET(M-1,N-1),C2C
2280   NEXT
2290 NEXT
2300 GOSUB *MSGCLS
2310 GOTO *LOOP
2320
2330
2340 *X.HANTEN
2350 FOR N=1 TO MY
2360   FOR M=1 TO MX/2
2370     LOCATE 23,N
2380     C1=POINT(180+M*8,N*8):C2=POINT(188+(MX-M)*8,N*8)
2390     T1=C1
2400     IF C1=7 AND POINT(180+M*8,N*8+1)=0 THEN C1=8:T1=0
2410     T2=C2
2420     IF C2=7 AND POINT(188+(MX-M)*8,N*8+1)=0 THEN C2=8:T2=0
2430     PUT@(184+(MX-M)*8,N*8),CLR%(C1*13),PSET
2440     PSET(MX-M,N-1),T1
2450     PUT@(176+M*8,N*8),CLR%(C2*13),PSET
2460     PSET(M-1,N-1),T2
2470   NEXT
2480 NEXT
2490 GOTO *LOOP
2500
2510
2520 *Y.HANTEN
2530 FOR N=1 TO MX
2540   FOR M=1 TO MY/2
2550     LOCATE 22+N,1
2560     C1=POINT(180+N*8,M*8):C2=POINT(180+N*8,(MY-M+1)*8)
2570     T1=C1
2580     IF C1=7 AND POINT(180+N*8,M*8+1)=0 THEN C1=8:T1=0
2590     T2=C2
2600     IF C2=7 AND POINT(180+N*8,(MY-M+1)*8+1)=0 THEN C2=8:T2=0
2610     PUT@(176+N*8,(MY-M+1)*8),CLR%(C1*13),PSET
2620     PSET(N-1,MY-M),T1
2630     PUT@(176+N*8,M*8),CLR%(C2*13),PSET
2640     PSET(N-1,M-1),T2
2650   NEXT
2660 NEXT
2670 GOTO *LOOP
2680
2690
2700 *SHIFT
2710 LOCATE 0,19:PRINT "ホウワ (2,4,6,8) ? ";
2720 A=USR(0):A$=INPUT$(1)
2730 DR=VAL(A$):LOCATE 16:PRINT DR
2740 PRINT "イトウスル トットス ? ";
2750 A=USR(0):A$=INPUT$(1)
2760 DF=VAL(A$):LOCATE 16:PRINT DF
2770 IF DR=2 THEN *DOWN
2780 IF DR=4 THEN *LEFT

```



```

2790 IF DR=6 THEN *RIGHT
2800 IF DR=8 THEN *UE
2810 GOSUB *MSGCLS
2820 GOTO *LOOP
2830 '
2840 *DOWN
2850 IF DF<1 OR DF>MY-1 THEN 2990
2860 FOR N=1 TO MX
2870   FOR M=1 TO MY
2880     A=USR(0):LOCATE 22+N,MY-M+1
2890     IF MY-DF-M+1<=0 THEN GOTO 2960
2900     C1=POINT(180+N*8,(MY-DF-M+1)*8)
2910     T1=C1
2920     IF C1=7 AND POINT(180+N*8,(MY-DF-M+1)*8+1)=0 THEN C1=8:T1=0
2930     PUT@(176+N*8,(MY-M+1)*8),CLR%(C1*13),PSET
2940     PSET(N-1,MY-M),T1
2950     GOTO 2970
2960     PUT@(176+N*8,(MY-M+1)*8),CLR%(0),PSET:PSET(N-1,MY-M),0
2970   NEXT
2980 NEXT
2990 GOSUB *MSGCLS
3000 GOTO *LOOP
3010 '
3020 *LEFT
3030 IF DF<1 OR DF>MX-1 THEN 3160
3040 FOR N=1 TO MY
3050   FOR M=1 TO MX
3060     A=USR(0):LOCATE 22+M,N
3070     IF MX-DF-M+1<=0 THEN GOTO 3130
3080     C1=POINT(180+(DF+M)*8,N*8)
3090     T1=C1
3100     IF C1=7 AND POINT(180+(DF+M)*8,N*8+1)=0 THEN C1=8:T1=0
3110     PUT@(176+M*8,N*8),CLR%(C1*13),PSET:PSET(M-1,N-1),T1
3120     GOTO 3140
3130     PUT@(176+M*8,N*8),CLR%(0),PSET:PSET(M-1,N-1),0
3140   NEXT
3150 NEXT
3160 GOSUB *MSGCLS
3170 GOTO *LOOP
3180 '
3190 *RIGHT
3200 IF DF<1 OR DF>MX-1 THEN 3330
3210 FOR N=1 TO MY
3220   FOR M=MX TO 1 STEP -1
3230     A=USR(0):LOCATE 22+M,N
3240     IF M<=DF THEN 3300
3250     C1=POINT(180+(M-DF)*8,N*8)
3260     T1=C1
3270     IF C1=7 AND POINT(180+(M-DF)*8,N*8+1)=0 THEN C1=8:T1=0
3280     PUT@(176+M*8,N*8),CLR%(C1*13),PSET:PSET(M-1,N-1),T1
3290     GOTO 3310
3300     PUT@(176+M*8,N*8),CLR%(0),PSET:PSET(M-1,N-1),0
3310   NEXT
3320 NEXT
3330 GOSUB *MSGCLS
3340 GOTO *LOOP
3350 '
3360 *UE
3370 IF DF<1 OR DF>MY-1 THEN 3500
3380 FOR N=1 TO MX
3390   FOR M=1 TO MY
3400     A=USR(0):LOCATE 22+N,M
3410     IF MY-DF-M+1<=0 THEN 3470
3420     C1=POINT(180+N*8,(M+DF)*8)

```

```

3430     T1=C1
3440     IF C1=7 AND POINT(180+N*8,(M+DF)*8+1)=0 THEN C1=8:T1=0
3450     PUT@(176+N*8,M*8).CLR%(C1*13),PSET:PSET(N-1,M-1),T1
3460     GOTO 3480
3470     PUT@(176+N*8,M*8).CLR%(0),PSET:PSET(N-1,M-1),0
3480     NEXT
3490 NEXT
3500 GOSUB *MSGCLS
3510 GOTO *LOOP
3520 '
3530 '
3540 *PALLET
3550 B$='7':GOSUB *FROMTO
3560 IF C1<>9 THEN COLOR=(C1,C2)
3570 GOSUB *MSGCLS
3580 GOTO *LOOP
3590 '
3600 '
3610 *LARGE
3620 FOR N=1 TO MY
3630     FOR M=1 TO MX
3640         C1=POINT(M-1,N-1)
3650         A=USR(0):LOCATE 22+M,N
3660         PUT@(176+M*8,N*8).CLR%(C1*13),PSET
3670     NEXT
3680 NEXT
3690 GOTO *LOOP
3700 '
3710 '
3720 '
3730 *FROMTO
3740 LOCATE 0,19
3750 PRINT "FROM (0-";B$;" ) ? ";
3760 A=USR(0):A$=INPUT$(1)
3770 IF A$<"0" OR A$>B$ THEN 3840
3780 C1=VAL(A$):LOCATE 12:PRINT C1
3790 PRINT "TO (0-";B$;" ) ? ";
3800 A=USR(0):A$=INPUT$(1)
3810 IF A$<"0" OR A$>B$ THEN 3840
3820 C2=VAL(A$):LOCATE 12:PRINT C2
3830 GOTO 3850
3840 C1=9
3850 RETURN
3860 '
3870 '
3880 *MSGCLS
3890 FOR N=19 TO 21 : LOCATE 0,N:PRINT SPC(19); : NEXT
3900 RETURN
3910 '
3920 '
3930 '
3940 *MAKE.DATA
3950 CONSOLE 0,25:CLS:LOCATE 0,5
3960 PRINT "1. DATA=B...R...G..."
3970 PRINT "2. DATA=BRG,BRG,... "
3980 PRINT "3. DATA=TBRG,TBRG,... "
3990 PRINT:PRINT "Select (1-3) ? ";
4000 A=USR(0):A$=INPUT$(1)
4010 A=VAL(A$):IF A<1 OR A>3 THEN CLS:GOTO 1210
4020 IF A=1 THEN DEF USR1=&HBF06 ELSE DEF USR1=&HBF4A
4030 D1$=' ':D2$='B':D3$='R':D4$='G'
4040 '
4050 CLS:LOCATE 0,5
4060 IF A=1 THEN PRINT "DATA=1...2...3..."

```

```

4070 IF A=2 THEN PRINT "DATA=123,123,... "
4080 IF A=3 THEN PRINT "DATA=1234,1234,... " : D1$="T"
4100 LOCATE 0,7
4110 PRINT " 1. Data.1 = ";
4120 IF A=3 THEN PRINT D1$;"(FIX)" ELSE PRINT D2$
4130 PRINT " 2. Data.2 = ";
4140 IF A=3 THEN PRINT D2$ ELSE PRINT D3$
4150 PRINT " 3. Data.3 = ";
4160 IF A=3 THEN PRINT D3$ ELSE PRINT D4$
4170 IF A=3 THEN PRINT " 4. Data.4 = ";D4$
4180 '
4190 CONSOLE 12,13:CLS:PRINT "Change Data ? ";
4200 C=USR(0):A$=INPUT$(1)
4210 IF A$="0" OR A$=CHR$(&HD) THEN 4450 ELSE B=VAL(A$) : CLS
4220 IF B=1 AND A<>3 THEN PRINT "Data.1 = ";:GOTO 4280
4230 IF B=2 THEN PRINT "Data.2 = ";:GOTO 4280
4240 IF B=3 THEN PRINT "Data.3 = ";:GOTO 4280
4250 IF B=4 AND A=3 THEN PRINT "Data.4 = ";:GOTO 4280
4260 GOTO 4190
4270 '
4280 C=USR(0):D$=INPUT$(1)
4290 IF D$=CHR$(&HD) OR D$="0" THEN D$=" ":D=0:GOTO 4350
4300 IF D$="B" OR D$="b" THEN D$="B":D=&H5C:GOTO 4350
4310 IF D$="R" OR D$="r" THEN D$="R":D=&H5D:GOTO 4350
4320 IF D$="G" OR D$="g" THEN D$="G":D=&H5E:GOTO 4350
4330 GOTO 4190
4340 '
4350 IF A<>3 THEN 4390
4360 IF B=2 THEN D2$=D$:IF D<>0 THEN POKE &HBF6E,D
4370 IF B=3 THEN D3$=D$:IF D<>0 THEN POKE &HBF80,D
4380 GOTO 4420
4390 IF B=1 THEN D2$=D$:IF D<>0 THEN POKE &HBF15,D:POKE &HBF6E,D
4400 IF B=2 THEN D3$=D$:IF D<>0 THEN POKE &HBF20,D:POKE &HBF80,D
4410 IF B=3 THEN D4$=D$:IF D<>0 THEN POKE &HBF2B,D:POKE &HBF8B,D
4420 IF B=4 THEN D4$=D$:IF D<>0 THEN POKE &HBF8B,D
4430 GOTO 4100
4440 '
4450 A=0
4460 IF D1$<>" " THEN A=A+1:POKE &HBF02,1
4470 IF D2$<>" " THEN A=A+1:POKE &HBF03,1
4480 IF D3$<>" " THEN A=A+1:POKE &HBF04,1
4490 IF D4$<>" " THEN A=A+1:POKE &HBF05,1
4500 B=USR1(0)
4510 CLS:PRINT "DATA ADDRESS":PRINT
4520 PRINT "B500H - ";HEX$(&HB4FF+((MX-1)*8+1)*MY*A);"H"
4530 '
4540 FOR N=0 TO 7 : COLOR=(N,N) : NEXT
4550 CONSOLE 0,25
4560 END
4570 '
4580
4590 DATA 10,04,00,00,00,00,F3,ED,4B,00,BF,11,00,B5,3A,03
4600 DATA BF,B7,28,05,D3,5C,CD,33,BF,3A,04,BF,B7,28,05,D3
4610 DATA 5D,CD,33,BF,3A,05,BF,B7,28,05,D3,5E,CD,33,BF,D3
4620 DATA 5F,FB,C9,21,00,C0,C5,C5,E5,7E,12,23,13,10,FA,E1
4630 DATA 01,50,00,09,C1,0D,20,EF,C1,C9,F3,ED,4B,00,BF,C5
4640 DATA 21,80,02,11,08,00,AF,ED,52,10,FB,EB,C1,21,87,C3
4650 DATA D9,ED,4B,00,BF,11,00,B5,21,00,C0,C5,E5,D3,5C,3A
4660 DATA 02,BF,B7,C4,A5,BF,3A,03,BF,B7,28,03,7E,12,13,D3
4670 DATA 5D,3A,04,BF,B7,28,03,7E,12,13,D3,5E,3A,05,BF,B7
4680 DATA 28,03,7E,12,13,23,10,D5,E1,01,50,00,09,C1,0D,20
4690 DATA CA,D3,5F,FB,C9,D9,AF,88,20,05,19,ED,4B,00,BF,C5
4700 DATA 06,08,7E,FE,82,28,03,AF,18,01,37,CB,11,23,10,F2
4710 DATA 79,C1,05,D9,12,13,C9

```


2 マップ・エディタ

マップ・エディタ

maped

```

10000 '***** Map Editor for SKY BRUISER *****
10010 CLEAR,&HBCFF:DEFINT A-Z:WIDTH 80,25:CONSOLE 0,25,0,0
10020 COLOR 0:SCREEN 0,1:CLS 3:PRINT "Working...!!"
10030 FOR I=0 TO 625 : READ A$:POKE &HBD00+I,VAL("&H"+A$) : NEXT
10040 '
10050 GOSUB *INIT
10060 '
10070 '
10080 '
10090 *MAIN
10100 CLS:CALL SIDECLS
10110 LOCATE 62,17:PRINT "Map Editor";
10120 LOCATE 62,19:PRINT "1. MAP PATTERN 1";
10130 LOCATE 62,20:PRINT "2. MAP PATTERN 2";
10140 LOCATE 62,21:PRINT "3. TEKI PATTERN";
10150 LOCATE 62,22:PRINT "4. LOAD,SAVE";
10160 LOCATE 62,23:PRINT "5. EXIT";
10170 LOCATE 62,24:PRINT "Which?";
10180 A$=INPUT$(1)
10190 ON INSTR("12345",A$) GOSUB *MAP1,*MAP2,*CHR,*COMM,*QUIT
10200 GOTO *MAIN
10210 '
10220 '
10230 '
10240 *QUIT
10250 LOCATE 0,24:PRINT "SURE? (y/n)";
10260 A$=INPUT$(1)
10270 IF A$="N" OR A$="n" THEN RETURN
10280 IF A$<>"Y" AND A$<>"y" THEN *QUIT
10290 FOR I=0 TO 7 : COLOR=(I,I) : NEXT
10300 END
10310 '
10320 '
10330 '
10340 *MAP1
10350 MAP1=1:GOTO *PUTMAP
10360 *MAP2
10370 MAP1=0
10380 '
10390 *PUTMAP
10400 GOSUB *MAPINIT
10410 *PUTMAP.LOOP
10420 GOSUB *CURSOR
10430 A$=INKEY$
10440 IF A$="" THEN *PUTMAP.LOOP
10450 IF A$="*" THEN RETURN
10460 IF A$=CHR$(127) THEN GOSUB *DEL:GOTO *PUTMAP.LOOP
10470 IF A$=CHR$(18) THEN GOSUB *INS:GOTO *PUTMAP.LOOP
10480 A=ASC(A$)
10490 IF A<97 OR A>122 THEN *PUTMAP.LOOP
10500 IF MAP1 THEN A=A-97 ELSE A=A-71
10510 POKE &HBFFF,A
10520 GOSUB *VAL.POKE
10530 CALL PUTMAP
10540 CALL KBC

```

```

10550 GOTO *PUTMAP.LOOP
10560 /
10570 /
10580 *INS
10590 IF LINE.END=LINE.MAX THEN 10650
10600 LINE.END=LINE.END+1
10610 GOSUB *VAL.POKE
10620 CALL INS
10630 CALL DISP
10640 CALL KBC
10650 RETURN
10660 /
10670 /
10680 *DEL
10690 IF CURSOR.LINE=LINE.END THEN 10750
10700 LINE.END=LINE.END-1
10710 GOSUB *VAL.POKE
10720 CALL DEL
10730 CALL DISP
10740 CALL KBC
10750 RETURN
10760 /
10770 /
10780 *MAPINIT
10790 CLS
10800 IF MAP1 THEN CALL MAPINI1 ELSE CALL MAPINI2
10810 LOCATE 59,0:PRINT "SET MAP PATTERN ";
10820 IF MAP1 THEN PRINT "1"; ELSE PRINT "2";
10830 FOR I=0 TO 25
10840   LOCATE 55+(5*I MOD 25),2+3*(I#5)
10850   PRINT CHR$(97+I);
10860 NEXT
10870 LOCATE 0,24
10880 COLOR 4
10890 PRINT "EXIT:*SET:a-zINS:INSDEL:DELMOVE:1-9";
10900 PRINT "-10:ROLL.UP+10:ROLL.DOWN";
10910 COLOR
10920 RETURN
10930 /
10940 /
10950 /
10960 *CHR
10970 GOSUB *CHRINIT
10980 *PUTCHR.LOOP
10990 GOSUB *CURSOR
11000 A$=INKEY$
11010 IF A$="" THEN *PUTCHR.LOOP
11020 IF A$="*" THEN RETURN
11030 IF A$=CHR$(127) THEN GOSUB *DEL:GOTO *PUTCHR.LOOP
11040 IF A$=CHR$(18) THEN GOSUB *INS:GOTO *PUTCHR.LOOP
11050 A=ASC(A$):IF A=32 THEN A=0:GOTO 11080
11060 IF A<97 OR A>122 THEN *PUTCHR.LOOP
11070 A=A-96
11080 POKE &HBFFF,A
11090 GOSUB *VAL.POKE
11100 CALL PUTCHR
11110 CALL KBC
11120 GOTO *PUTCHR.LOOP
11130 /
11140 /
11150 *CHRINIT
11160 CLS
11170 CALL CHRINI

```

```

11180 LOCATE 59,0:PRINT "SET TEKI PATTERN";
11190 FOR I=0 TO 25
11200   LOCATE 55+(5*I MOD 25),2+3*(I%5)
11210   PRINT CHR$(97+I);
11220 NEXT
11230 LOCATE 0,24
11240 COLOR 4
11250 PRINT "EXIT:*SET:a-zRESET:SPACEINS:INSDEL:DEL";
11260 PRINT "MOVE:1-9-10:ROLL.UP-10:ROLL.DOWN";
11270 COLOR 0
11280 RETURN
11290 '
11300 '
11310 '
11320 *COMM
11330 CONSOLE 24,1
11340 COLOR 4
11350 PRINT "EXIT:*LOAD DATA:lSAVE DATA:s";
11360 COLOR 0
11370 A$=INKEY$
11380 IF A$="" THEN 11370
11390 PRINT
11400 IF A$="*" THEN CONSOLE 0,25:RETURN
11410 IF A$="l" THEN *LOAD.MAP
11420 IF A$="s" THEN *SAVE.MAP
11430 GOTO *COM.
11440 '
11450 '
11460 *LOAD.MAP
11470 INPUT "load file name (RETURN:EXIT) = ",FILE$
11480 IF FILE$="" THEN RETURN
11490 BLOAD FILE$
11500 CURSOR.X=PEEK(&HBFF0)
11510 CURSOR.Y=PEEK(&HBFF1)
11520 CURSOR.LINE=PEEK(&HBFF2)+PEEK(&HBFF3)*256
11530 MAP.TOP=PEEK(&HBFF4)+PEEK(&HBFF5)*256
11540 LINE.END=PEEK(&HBFF6)+PEEK(&HBFF7)*256
11550 CALL DISP:CONSOLE 0,25
11560 RETURN
11570 '
11580 '
11590 *SAVE.MAP
11600 INPUT "save file name (RETURN:EXIT) = ",FILE$
11610 IF FILE$="" THEN RETURN
11620 BSAVE FILE,&HBFF0,(LINE.END+1)*26+16
11630 CONSOLE 0,25
11640 RETURN
11650 '
11660 '
11670 '
11680 *CURSOR
11690 GOSUB *NOTICE
11700 FLUSH=(FLUSH+1) MOD 6
11710 LOCATE CURSOR.X*4,CURSOR.Y*2
11720 IF FLUSH<3 THEN PRINT PUT,CURSOR$; ELSE PRINT ERASE.CURSOR$;
11730 A=INP(0):B=INP(1):C=INP(11)
11740 IF A=255 AND B=255 AND C=255 THEN RETURN
11750 CALL KBC
11760 IF A=253 THEN *DL
11770 IF A=251 THEN *D
11780 IF A=247 THEN *DR
11790 IF A=239 THEN *L
11800 IF A=191 THEN *R

```



```

11810 IF A=127 THEN *UL
11820 IF B=254 THEN *U
11830 IF B=253 THEN *UR
11840 IF C=254 THEN *ROLL.UP
11850 IF C=253 THEN *ROLL.DOWN
11860 RETURN
11870 '
11880 *DL
11890 IF CURSOR.X=0 OR CURSOR.LINE=0 THEN 11930
11900 IF CURSOR.Y=11 THEN MAP.TOP=MAP.TOP-1:GOSUB *VAL.POKE:CALL DISP
11910 CURSOR.LINE=CURSOR.LINE-1
11920 DX=-1:DY=-(CURSOR.Y<>11):GOSUB *MOVE.CURSOR
11930 RETURN
11940 '
11950 *D
11960 IF CURSOR.LINE=0 THEN 12000
11970 IF CURSOR.Y=11 THEN MAP.TOP=MAP.TOP-1:GOSUB *VAL.POKE:CALL DISP
11980 CURSOR.LINE=CURSOR.LINE-1
11990 DX=0:DY=-(CURSOR.Y<>11):GOSUB *MOVE.CURSOR
12000 RETURN
12010 '
12020 *DR
12030 IF CURSOR.X=12 OR CURSOR.LINE=0 THEN 12070
12040 IF CURSOR.Y=11 THEN MAP.TOP=MAP.TOP-1:GOSUB *VAL.POKE:CALL DISP
12050 CURSOR.LINE=CURSOR.LINE-1
12060 DX=1:DY=-(CURSOR.Y<>11):GOSUB *MOVE.CURSOR
12070 RETURN
12080 '
12090 *L
12100 IF CURSOR.X=0 THEN 12120
12110 DX=-1:DY=0:GOSUB *MOVE.CURSOR
12120 RETURN
12130 '
12140 *R
12150 IF CURSOR.X=12 THEN 12170
12160 DX=1:DY=0:GOSUB *MOVE.CURSOR
12170 RETURN
12180 '
12190 *UL
12200 IF CURSOR.X=0 OR CURSOR.LINE=LINE.MAX THEN 12270
12210 IF CURSOR.LINE<>LINE.END THEN 12240
12220 LINE.END=LINE.END+1:GOSUB *VAL.POKE:CALL ADDLINE
12230 IF CURSOR.Y THEN CALL DISP
12240 IF CURSOR.Y=0 THEN MAP.TOP=MAP.TOP+1:GOSUB *VAL.POKE:CALL DISP
12250 CURSOR.LINE=CURSOR.LINE+1
12260 DX=-1:DY=(CURSOR.Y<>0):GOSUB *MOVE.CURSOR
12270 RETURN
12280 '
12290 *U
12300 IF CURSOR.LINE=LINE.MAX THEN 12370
12310 IF CURSOR.LINE<>LINE.END THEN 12340
12320 LINE.END=LINE.END+1:GOSUB *VAL.POKE:CALL ADDLINE
12330 IF CURSOR.Y THEN CALL DISP
12340 IF CURSOR.Y=0 THEN MAP.TOP=MAP.TOP+1:GOSUB *VAL.POKE:CALL DISP
12350 CURSOR.LINE=CURSOR.LINE+1
12360 DX=0:DY=(CURSOR.Y<>0):GOSUB *MOVE.CURSOR
12370 RETURN
12380 '
12390 *UR
12400 IF CURSOR.X=12 OR CURSOR.LINE=LINE.MAX THEN 12470
12410 IF CURSOR.LINE<>LINE.END THEN 12440
12420 LINE.END=LINE.END+1:GOSUB *VAL.POKE:CALL ADDLINE
12430 IF CURSOR.Y THEN CALL DISP

```

```

12440 IF CURSOR.Y=0 THEN MAP.TOP=MAP.TOP+1:GOSUB *VAL.POKE:CALL DISP
12450 CURSOR.LINE=CURSOR.LINE+1
12460 DX=1:DY=(CURSOR.Y<>0):GOSUB *MOVE.CURSOR
12470 RETURN
12480 '
12490 '
12500 *MOVE.CURSOR
12510 LOCATE CURSOR.X*4,CURSOR.Y*2
12520 PRINT ERASE.CURSOR$;
12530 CURSOR.X=CURSOR.X+DX:CURSOR.Y=CURSOR.Y+DY
12540 LOCATE CURSOR.X*4,CURSOR.Y*2
12550 PRINT PUT.CURSOR$;:GOSUB *VAL.POKE
12560 RETURN
12570 '
12580 '
12590 *ROLL.UP
12600 IF MAP.TOP<10 THEN CURSOR.LINE=CURSOR.LINE-MAP.TOP:MAP.TOP=0:GOTO 12620
12610 MAP.TOP=MAP.TOP-10:CURSOR.LINE=CURSOR.LINE-10
12620 GOSUB *VAL.POKE:CALL DISP
12630 RETURN
12640 '
12650 '
12660 *ROLL.DOWN
12670 IF CURSOR.LINE+10>LINE.END THEN RETURN
12680 MAP.TOP=MAP.TOP+10:CURSOR.LINE=CURSOR.LINE+10
12690 GOSUB *VAL.POKE:CALL DISP
12700 RETURN
12710 '
12720 '
12730 '
12740 '
12750 *INIT
12760 LINE.MAX=370
12770 MAP.TOP=0
12780 LINE.END=0
12790 CURSOR.X=0
12800 CURSOR.Y=11
12810 CURSOR.LINE=0
12820 FLUSH=0
12830 '
12840 INIT=&HBD00
12850 DISP=&HBD09
12860 PUTMAP=&HBD03
12870 PUTCHR=&HBD06
12880 INS=&HBD0F
12890 DEL=&HBD12
12900 ADDLINE=&HBD0C
12910 MAPINI1=&HBD15
12920 MAPINI2=&HBD18
12930 CHRINI=&HBD1B
12940 SIDECLS=&HBD1E
12950 KBC=&H35D9
12960 '
12970 DN$=CHR$(31):LE$=CHR$(29)
12980 PUT.CURSOR$=" " + DN$ + LE$ + LE$ + LE$ + LE$ + " "
12990 ERASE.CURSOR$=" " + DN$ + LE$ + LE$ + LE$ + LE$ + " "
13000 COLOR=(0,1),(1,4),(2,2),(3,2),(4,0),(5,0),(6,7),(7,7)
13010 CALL INIT
13020 GOSUB *VAL.POKE:CALL ADDLINE:CALL DISP
13030 RETURN
13040 '
13050 '
13060 '

```

```

13070 *VAL.POKE
13080 POKE &HBFF0,CURSOR.X
13090 POKE &HBFF1,CURSOR.Y
13100 POKE &HBFF2,CURSOR.LINE MOD 256
13110 POKE &HBFF3,CURSOR.LINE * 256
13120 POKE &HBFF4,MAP.TOP MOD 256
13130 POKE &HBFF5,MAP.TOP * 256
13140 POKE &HBFF6,LINE.END MOD 256
13150 POKE &HBFF7,LINE.END * 256
13160 RETURN
13170
13180
13190
13200 *NOTICE
13210 LOCATE 62,18:PRINT "cursor.x      = ";CURSOR.X
13220 LOCATE 62,19:PRINT "cursor.y      = ";CURSOR.Y
13230 LOCATE 62,20:PRINT "cursor.line   = ";CURSOR.LINE
13240 LOCATE 62,21:PRINT "map.top       = ";MAP.TOP
13250 LOCATE 62,22:PRINT "line.end      = ";LINE.END;
13260 RETURN
13270
13280
13290
13300 DATA C3,E7,BD,C3,04,BE,C3,2B,BE,C3,57,BE,C3,C8,BE,C3
13310 DATA D6,BE,C3,F8,BE,C3,41,BF,C3,3D,BF,C3,50,BF,C3,60
13320 DATA BF,CD,95,BD,CD,A7,BD,D3,5C,CD,42,BD,D3,5F,C9,CD
13330 DATA 95,BD,CD,B5,BD,D3,5D,CD,42,BD,D3,5E,CD,42,BD,D3
13340 DATA 5F,C9,ED,73,5E,BD,31,4C,00,ED,58,61,BD,01,FF,10
13350 DATA ED,A0,ED,A0,ED,A0,ED,A0,EB,39,EB,10,F3,31,00,00
13360 DATA C9,00,00,22,93,BD,CD,95,BD,AF,D3,5C,CD,7C,BD,D3
13370 DATA 5D,CD,7C,BD,D3,5E,CD,7C,BD,D3,5F,C9,2A,61,BD,11
13380 DATA 50,00,ED,4B,93,BD,C5,E5,77,23,10,FC,E1,19,C1,0D
13390 DATA 20,F4,C9,00,00,68,26,00,29,29,29,29,29,29,09,01
13400 DATA 00,C0,09,22,61,BD,C9,11,00,70,6F,26,00,29,29,29
13410 DATA 29,29,29,19,C9,11,00,60,6F,26,00,29,18,EF,D5,C5
13420 DATA 29,4D,44,29,29,5D,54,29,09,19,11,00,C0,19,C1,D1
13430 DATA C9,F3,21,C2,E6,7E,F6,02,77,D3,31,C9,21,C2,E6,7E
13440 DATA E6,FD,77,D3,31,FB,C9,21,00,60,06,80,AF,77,23,10
13450 DATA FC,C9,2A,F0,BF,7D,87,87,4F,7C,87,87,47,3A,FF,BF
13460 DATA CD,21,BD,C9,CD,D1,BD,21,21,BD,22,01,BE,CD,F2,BD
13470 DATA 2A,F2,BF,CD,BE,BD,3A,F0,BF,87,5F,16,00,19,7E,E6
13480 DATA 80,5F,3A,FF,BF,B3,77,CD,DC,BD,C9,CD,D1,BD,21,2F
13490 DATA BD,22,01,BE,CD,F2,BD,2A,F2,BF,CD,BE,BD,3A,F0,BF
13500 DATA 87,5F,16,00,19,3A,FF,BF,B7,28,04,CB,FE,18,02,CB
13510 DATA BE,23,77,CD,DC,BD,C9,CD,D1,BD,AF,32,C7,BE,ED,5B
13520 DATA F4,BF,2A,F6,BF,ED,52,7D,2E,0C,24,25,20,0D,FE,0B
13530 DATA 30,09,6F,2C,D6,0B,ED,44,32,C7,BE,EB,CD,BE,BD,01
13540 DATA 00,2C,16,0D,D5,C5,E5,7E,E6,7F,CD,21,BD,E1,C1,23
13550 DATA C5,E5,7E,CD,2F,BD,E1,C1,D1,23,79,C6,04,4F,15,20
13560 DATA E3,4A,78,D6,04,47,1D,20,D9,CD,DC,BD,3A,C7,BE,B7
13570 DATA C8,F3,87,87,87,87,6F,26,34,22,93,BD,21,00,C0,22
13580 DATA 61,BD,CD,69,BD,FB,C9,00,2A,F6,BF,CD,BE,BD,06,1A
13590 DATA AF,77,23,10,FC,C9,2A,F2,BF,CD,BE,BD,E5,EB,2A,F6
13600 DATA BF,CD,BE,BD,E5,B7,ED,52,4D,44,E1,2B,E5,11,1A,00
13610 DATA 19,EB,E1,ED,B8,E1,18,D6,2A,F2,BF,CD,BE,BD,EB,2A
13620 DATA F6,BF,23,CD,BE,BD,B7,ED,52,4D,44,21,1A,00,19,ED
13630 DATA 80,C9,CD,D1,BD,01,37,06,1E,00,D5,C5,78,CD,21,BD
13640 DATA C1,D1,1C,79,C6,05,4F,FE,4C,38,EF,0E,37,78,C6,06
13650 DATA 47,FE,22,38,E5,78,CD,21,BD,CD,DC,BD,C9,3E,1A,18
13660 DATA 01,AF,32,19,BF,21,21,BD,22,1E,BF,22,37,BF,18,C2
13670 DATA 3E,01,32,19,BF,21,2F,BD,22,1E,BF,22,37,BF,18,B2
13680 DATA F3,21,C8,1C,22,93,BD,21,34,C0,22,61,BD,CD,69,BD
13690 DATA FB,C9

```


③ データ・ジェネレータ

データ・ジェネレータ

maker

```

10000 '***** Data Maker for SKY BRUISER *****
10010 CLEAR,&HBEFF:DEFINT A-Z:SCREEN 0,2:CLS:PRINT "Working...!!"
10020 FOR I=0 TO 198:READ A:POKE &HBF00+I,VAL("&H"+A$):NEXT
10030 '
10040 INIT=&HBF00
10050 APPEND=&HBF03
10060 FINISH=&HBF06
10070 MAP.MAX=&H4FFF
10080 EM.TAIL=&H5FFF
10090 CLS
10100 CALL INIT
10110 '
10120 '
10130 '
10140 *MAIN
10150 PRINT
10160 PRINT "          1. GENERATE DATA (APPEND DATA)"
10170 PRINT "          2. END"
10180 PRINT
10190 PRINT "          which ?";
10200 ON INSTR("12",INPUT$(1)) GOSUB *APPEND,*QUIT
10210 GOTO *MAIN
10220 '
10230 '
10240 '
10250 *APPEND
10260 PRINT:PRINT
10270 IF EM.TAIL=&H8000 THEN BEEP:PRINT "Out of memory":RETURN
10280 INPUT "file name (RETURN:EXIT) = ",FILE$
10290 IF FILE$="" THEN RETURN
10300 BLOAD FILE$
10310 LINE.END=PEEK(&HBFF6)+PEEK(&HBFF7)*256
10320 MAP.TAIL=PEEK(&HBFE0)+PEEK(&HBFE1)*256
10330 IF MAP.TAIL+LINE.END*13+182>MAP.MAX THEN BEEP:PRINT "Out of memory":RETURN
10340 CALL APPEND
10350 GOSUB *GET.PRINT
10360 IF EM.TAIL=&H8000 THEN BEEP:PRINT "Out of memory"
10370 RETURN
10380 '
10390 '
10400 '
10410 *QUIT
10420 PRINT
10430 CALL FINISH
10440 GOSUB *GET.PRINT
10450 PUT.TOP=PEEK(&HBFE4)+PEEK(&HBFE5)*256
10460 DAT.END=PEEK(&HBFE6)+PEEK(&HBFE7)*256
10470 PRINT
10480 INPUT "save file name = ",FILE$
10490 BSAVE FILE$,&H1000,DAT.END-&H1000
10500 PRINT
10510 PRINT "SCENT(Top of map)   = &H";RIGHT$("00"+HEX$(PUT.TOP-&HFFF),4)
10520 PRINT "EMENT(Top of enemy) = &H";RIGHT$("00"+HEX$(PUT.TOP+183-&H1000),4)
10530 PRINT
10540 PRINT "saved from &H0000 to &H";RIGHT$("00"+HEX$(DAT.END-&HFFF),4)

```

```

10550 FIELD #1,2 AS START$,2 AS END.$
10560 OPEN FILE$ AS #1
10570 GET #1
10580 TEMP$=MK!$(CVI(START$)-&H1000):LSET START$=TEMP$
10590 TEMP$=MK!$(CVI(END.$)-&H1000):LSET END.$=TEMP$
10600 PUT #1,1
10610 CLOSE #1
10620 SCREEN ,0
10630 END
10640 '
10650 '
10660 '
10670 *GET,PRINT
10680 MAP.TAIL=PEEK(&HBFE0)+PEEK(&HBFE1)*256
10690 EM.TAIL=PEEK(&HBFE2)+PEEK(&HBFE3)*256
10700 PRINT
10710 PRINT 'Map data   = &H1000 - &H';HEX$(MAP.TAIL-1)
10720 PRINT 'Enemy data = &H6000 - &H';HEX$(EM.TAIL-1)
10730 RETURN
10740 '
10750 '
10760 '
10770 DATA C3,18,BF,C3,6D,BF,C3,2F,BF,D5,C5,29,4D,44,29,29
10780 DATA 5D,54,29,19,09,C1,D1,C9,F3,21,00,10,3E,03,06,B6
10790 DATA 77,23,10,FC,22,E0,BF,21,00,60,22,E2,BF,FB,C9,F3
10800 DATA 2A,E0,BF,2B,22,E4,BF,23,3E,03,06,B6,77,23,10,FC
10810 DATA 22,E0,BF,3A,C2,E6,F6,02,D3,31,2A,E2,BF,11,00,60
10820 DATA ED,52,4D,44,2A,E0,BF,22,E6,BF,28,0A,EB,21,00,60
10830 DATA ED,B0,ED,53,E6,BF,3A,C2,E6,D3,31,FB,C9,F3,2A,F6
10840 DATA BF,4D,44,03,CD,09,BF,11,00,C0,19,ED,5B,E0,BF,C5
10850 DATA 06,0D,7E,23,23,12,13,10,F9,D5,11,CC,FF,19,D1,C1
10860 DATA 0B,78,B1,20,EA,ED,53,E0,BF,11,01,C0,2A,E2,BF,ED
10870 DATA 4B,F6,BF,03,C5,06,0D,1A,13,13,B7,28,07,77,23,7C
10880 DATA B7,FA,C1,BF,10,F1,C1,0B,78,B1,20,E8,22,E2,BF,FB
10890 DATA C9,C1,22,E2,BF,FB,C9

```

4. マシン語命令小辞典…御一読アレッと！

ここにある説明は、あくまでも用語の解説だけであり、使用可能なレジスタとかフラグの変化などの細かい利用法については、Appendix2のマシン語インストラクション一覧表を参照してください。また、I/Oポートの詳しい内容については、参考書『PC-Techknow 8800』等に出ていますので、それを見てください。

- ADC** : Add with Carry の略。レジスタとレジスタ、数値、レジスタで示される番地の内容と、キャリーフラグの値を加算する。
- ADD** : レジスタとレジスタ、数値、レジスタで示される番地の内容を加算する。
- AND** : アキュムレータとレジスタ、数値、レジスタで示される番地の内容との論理積 (AND) を取り、その値はアキュムレータに入る。論理積とは、双方をビット毎に見て、両方のビットが共に1ならばそのビットを1にし、それ以外の時は0にするという演算で、一般にアキュムレータの特定のビットをかならずりにしたい場合に使われる。
- BIT** : レジスタまたはレジスタで示される番地の内容の、指定のビットが1か0か調べる。答はゼロフラグで返し、0の時はフラグをセットし、1の時にはフラグをアンセットする。
- CALL** : BASIC の GOSUB 命令と同じようなもので、指定の番地へ飛び、RET で戻ってくる。フラグ判定による条件付きの使い方もできる。
- CCF** : Complement Carry Flag の略。キャリーフラグの値を1なら0に、0なら1に反転する。
- CP** : Compare の略。アキュムレータとレジスタ、数値、レジスタで示される番地の内容とを比較する。結果はフラグに反映させるだけで、レジスタの値は変化しない。
- CPD** : Compare Decrement の略。アキュムレータと HL レジスタが示す番地の内容を比較し、結果をフラグに反映させる。HL レジスタ、BC レジスタは共に-1される。
- CPDR** : Compare Decrement Repeat の略。アキュムレータと HL レジスタが示す番地の内容が等しくなるか、または BC レジスタの値が0になるまで CPD を繰り返す。

- 返す。
- CPI** : Compare Increment の略。アキュムレータと HL レジスタが示す番地の内容を比較し、結果をフラグに反映させる。HL レジスタは+1され、BC レジスタは-1される。
- CPIR** : ComPare Increment Repeat の略。アキュムレータと HL レジスタが示す番地の内容が等しくなるか、または BC レジスタの値が0になるまで CPI を繰り返す。
- CPL** : ComPLement accumulator の略。アキュムレータの各ビットの内容を1なら0に、0なら1に反転する。
- DAA** : Decimal Adjust Accumulator の略。BCD 演算をする場合に、演算後のアキュムレータの値を補正する命令。BCD 演算については、3章を参照。
- DB** : Define Byte の略。アセンブラ用の擬似命令で、1バイト単位でデータの設定ができる。
- DEC** : DECrement の略。レジスタまたはレジスタで示される番地の内容を-1する。
- DI** : Disable Interrupts の略。割り込み禁止のこと。DI をすると、割り込みがかからない分だけ実行速度のアップにもなる。
- DJNZ** : Decrement Jump if Non Zero の略。B レジスタの値を-1し、0でなければ指定先へジャンプする。ジャンプできる範囲は相対ジャンプの範囲内(現在番地を基準にして-80_H~7F_H番地)に限られる。FOR~NEXT 文のようにループとして使える。
- DS** : Define Storage の略。アセンブラ用の擬似命令で、必要なだけフリー領域を取ることができるが、その内容は不定である。
- DW** : Define Word の略。アセンブラ用の擬似命令で、2バイト単位でデータの設定ができるが、メモリには上位・下位が逆になって入る。
- EI** : Enable Interrupts の略。割り込みの許可。
- END** : アセンブラ用の擬似命令で、プログラムの終了を意味する。この命令以降のプログラムはアセンブルされない。
- EQU** : EQUate の略。アセンブラ用の擬似命令で、ラベルに値を与える。
- EX** : EXchange の略。レジスタ間、またはレジスタとスタック間でデータの交換を

- する。
- EXX** : BC、DE、HL の各レジスタを、裏レジスタの BC'、DE'、HL' の内容と交換する。または、その逆。
- HALT** : プログラム実行の停止。
- IM** : Interrupt Mode の略。割り込みのモード設定をする命令で、IM0、IM1、IM2 の3つのモードがある。
- IN** : INput の略。指定した入力ポートから、1 バイトのデータをレジスタに取り入れる。入力ポートの詳細に関しては『PC-Techknow 8800』等を参照。
- INC** : INCrement の略。レジスタまたはレジスタで示される番地の内容を +1 する。
- IND** : INput Decrement の略。C レジスタが示す入力ポートのデータを、HL レジスタが示す番地に取り入れる。HL レジスタ、B レジスタの値は共に -1 される。
- INDR** : INput Decrement Repeat の略。B レジスタの値が 0 になるまで、IND を繰り返す。
- INI** : INput Increment の略。C レジスタが示す入力ポートのデータを、HL レジスタが示す番地に取り入れる。HL レジスタの値は +1 され、B レジスタの値は -1 される。
- INIR** : INput Increment Repeat の略。B レジスタの値が 0 になるまで、INI を繰り返す。
- JP** : Jump の略。指定の番地へジャンプする。フラグ判定による条件付きのジャンプもできる。
- JR** : Jump Rerative の略。相対ジャンプ命令といい、指定の番地へジャンプするが、指定できるのは、現在の番地を基準にして $-80_{\text{H}} \sim +7\text{F}_{\text{H}}$ の範囲に限られている。フラグ判定による条件付きの相対ジャンプ命令もできる。
- LD** : LoaD の略。レジスタに数値、レジスタの値、絶対番地で示される内容、レジスタで示される番地の内容を代入したり、その逆にレジスタの値、または数値をメモリに入れる。一番使用頻度が高い。
- LDD** : LoaD Decrement の略。HL レジスタで示される番地の内容を、DE レジスタで示される番地に入れる。HL、DE、BC レジスタの値はすべて -1 される。
- LDDR** : LoaD Decrement Repeat の略。BC レジスタの値が 0 になるまで、LDD を繰

- り返す。データのブロック転送に使われる。
- LDI** : Load Increment の略。HL レジスタで示される番地の内容を、DE レジスタで示される番地に入れる。HL、DE レジスタは+1され、BC レジスタは-1される。
- LDIR** : Load Increment Repeat の略。BC レジスタ値が0になるまで、LDI を繰り返す。データのブロック転送に使われる。
- NEG** : NEGate の略。CPL 命令を実行して1を加える。
- NOP** : No OPeration の略。何もしない。
- OR** : アキュムレータとレジスタ、数値、レジスタで示される番地の内容との論理和 (OR) を取り、その値はアキュムレータに入る。論理和とは、双方をビット毎に見て、どちらか一方でも1であればそのビットを1にし、それ以外は0にするという演算で一般にアキュムレータの特定のビットを必ず1にしたい場合に使われる。OR A (AND A) を CP 0 の代わりに用いることが多い。
- ORG** : ORiGin の略。アセンブラ用の疑似命令で、プログラムの開始番地を指定する。
- OTDR** : OuTput Decrement Repeat の略。B レジスタの値が0になるまで、OUTD を繰り返す。
- OTIR** : OuTput Increment Repeat の略。B レジスタの値が0になるまで、OUTI を繰り返す。
- OUT** : 指定の出力ポートに、レジスタの値を出力する。出力ポートの詳細については PC-Techknow 8800 等を参照。
- OUTD** : OuTput Decrement の略。HL レジスタが示す番地の内容を、C レジスタが示す出力ポートへ出力する。HL レジスタ、B レジスタの値は共に-1される。
- OUTI** : OuTput Increment の略。HL レジスタが示す番地の内容を、C レジスタが示す出力ポートへ出力する。HL レジスタの値は+1され、B レジスタの値-1される。
- POP** : スタック・エリアにあるデータをペアレジスタに取り入れる。スタック・ポインタは+2される。アキュムレータの場合はフラグ・レジスタとのペアになるため、フラグの変化がある。
- PUSH** : ペアレジスタの内容をスタック・エリアに退避する。スタック・ポインタは-2

される。アキュムレータは AF として、フラグ・レジスタの内容も退避される。

RES : RESet の略。レジスタまたはレジスタで示される番地の内容の、指定のビットの値を 1 にする。

RET : RETurn の略。サブルーチンからスタック・ポインタで示される番地に戻される。スタック・ポインタは +2 される。普通は CALL 命令と対で使用され、CALL の次の命令に戻る。フラグ判定による条件付きの復帰もできる。

RETI : RETurn from Interrupt の略。マスク可能割込み処理ルーチンから戻る。スタック・ポインタは +2 される。

RETN : RETurn from Non maskable interrupt の略。マスク不能割込み処理ルーチンから戻る。スタック・ポインタは +2 される。

RL : Rotate Left の略。レジスタまたはレジスタで示される番地の内容を左方向にビット・シフトする。最上位ビットの値はキャリーフラグに入り、最下位ビットにはキャリーフラグの値が入る。

RLA : Rotate Left Accumulator の略。アキュムレータ専用の RL 命令で、RL A とはフラグの変化、必要バイト数が違うだけである。

RLC : Rotate Left Circuler の略。レジスタまたはレジスタで示される番地の内容を左方向にビット・シフトする。最上位ビットの値がキャリーフラグと最下位ビットに入る。

RLCA : Rotate Left Circuler Accumulator の略。アキュムレータ専用の RLC 命令で、RLC A とはフラグの変化、必要バイト数が違うだけである。

RLD : Rotate Left Digit の略。HL レジスタが示す番地の内容の下位 4 ビットが上位 4 ビットに、上位 4 ビットはアキュムレータの下位 4 ビットに、下位 4 ビットにはアキュムレータの下位 4 ビットが入る。

RR : Rotate Right の略。レジスタまたはレジスタで示される番地の内容を右方向にビット・シフトする。最下位ビットの値はキャリーフラグに入り、最上位ビットにはキャリーフラグの値が入る。

RRA : Rotate Right Accumulator の略。アキュムレータ専用の RR 命令で、RR A とはフラグの変化、必要バイト数が違う。

RRC : Rotate Right Circuler の略。レジスタまたはレジスタで示される番地の内容を右方向にビット・シフトする。最下位ビットの値がキャリーフラグと最上位ビットに入る。

- RRCA** : Rotate Right Circular Accumulator の略。アキュムレータ専用の RRC 命令で、RRCA とはフラグの変化、必要バイト数が違う。
- RRD** : Rotate Right Digit の略。HL レジスタが示す番地の内容の上位 4 ビットが下位 4 ビットに、下位 4 ビットはアキュムレータの下位 4 ビットに、上位 4 ビットにはアキュムレータの下位 4 ビットが入る。
- RST** : ReSTart の略。リスタートは、本来割り込みモード 0 の時の処理ルーチン・アドレスであるが、分岐先の決まった CALL 命令と同じである。割り込みレベルは、 00_{H} 、 08_{H} 、 10_{H} 、 18_{H} 、 20_{H} 、 28_{H} 、 30_{H} 、 38_{H} の 8 種類があり、本書では $\text{RST } 38_{\text{H}} (= \text{CALL } 0038_{\text{H}})$ を、 h' の状態に戻る時に使っている。
- SBC** : SuBtract with Carry の略。レジスタからレジスタ、数値、レジスタで示される番地の内容と、キャリーフラグの値を減算する。
- SCF** : Set Carry Flag の略。キャリーフラグをセットする (1 にする)。
- SET** : レジスタまたはレジスタで示される番地の内容の、指定のビットだけを 1 にする。
- SLA** : Shift Left Arithmentic の略。レジスタまたはレジスタで示される番地の内容を左にビット・シフトする。最上位ビットの値はキャリーフラグに入り、最下位ビットには 0 が入る。
- SRA** : Shift Right Arithmetic の略。レジスタまたはレジスタで示される番地の内容を右にビット・シフトする。最上位ビットの値は変化がなく、最下位ビットの値はキャリーフラグに入る。
- SRL** : Shift Right Logical の略。レジスタまたはレジスタで示される番地の内容を右にビット・シフトする。最上位ビットには 0 が入り、最下位ビットの値はキャリーフラグに入る。
- SUB** : SUBtract の略。アキュムレータからレジスタ、数値、レジスタで示される番地の内容を減算する。
- XOR** : eXclusive OR の略。アキュムレータとレジスタ、数値、レジスタで示される番地の内容との排他的論理和 (XOR) を取り、その値はアキュムレータに入る。排他的論理和とは双方をビット毎に見て、互いに値の違っているビットだけを 1 にするという演算で、アキュムレータの特定のビットを反転 (1 なら 0 に、0 なら 1 に) する場合に使われる。また、アキュムレータを 0 にする手段として、 XOR A は多用されている。

参考文献

- ・ PC8801 N88-BASIC 解析マニュアル 川村清著(秀和システム・トレーディング)
- ・ PC-8801mkIISR テクニカルメモ アスキーHSP編(アスキー)
- ・ PC-Techknow 8800 システムソフト編(アスキー)
- ・ はじめて読むマシン語 村瀬康治著(アスキー)
- ・ はじめて読むアセンブラ 村瀬康治著(アスキー)
- ・ μ com-82 ユーザーズ・マニュアル(日本電気株式会社)

協力

大熊英男・石塚圭樹・藤井敬雄

PC-8801mk II SR

マシン語ゲーム・プログラミング

1985年10月25日 初版発行

1990年3月11日 第1版第8刷発行

定価2,580円 (本体2,505円)

著者 ひだか ちかお 日高 徹

発行者 塚本慶一郎

発行所 株式会社 **アスキー**

〒107 東京都港区南青山6-11-1スリーエフ南青山ビル

振替 東京4-161144

TEL (03)486-7111 (大代表)

情報TEL (03)498-0299 (ダイヤルイン)

出版営業部 TEL (03)486-1977 (ダイヤルイン)

本書は著作権法上の保護を受けています。本書の一部あるいは全部について(ソフトウェア及びプログラムを含む)、株式会社アスキーから文書による許諾を得ずに、いかなる方法においても無断で複写、複製することは禁じられています。

編集担当 竹山正寿

表紙担当 郷 啓子

印刷 株式会社加藤文明社印刷所

ISBN4-87148-166-2

PC 88
mk II
SR

マシン語ゲームプログラミング

日高 徹 著

ASCII
アスキー・パブリッシング

PC-8801mkII SR

マシン語ゲーム
プログラミング

日高 徹 著

CHAPTER **1**
ウォーミング・アップ

CHAPTER **2**
キャラクタ・パターンの表示と移動

CHAPTER **3**
衝突と得点計算

CHAPTER **4**
音楽演奏と効果音

CHAPTER **5**
迷路型ゲーム

CHAPTER **6**
スクロール・ゲーム

APPENDIX
MF-ASM2
インストラクション表
ツール
マシン語命令小辞典

定価2,580円
(本体2,505円)

ISBN4-87148-166-2 C3055 P2580E